ICSEA 2013

The Eighth International Conference on Software Engineering Advances

ISBN: 978-1-61208-304-9

October 27 - November 1, 2013

Venice, Italy

ICSEA 2013 Editors

Luigi Lavazza, Università dell'Insubria - Varese, Italy

Roy Oberhauser, Aalen University, Germany

Adriana Martin, National University of Austral Patagonia (UNPA), Argentina

Jameleddine Hassine, KFUPM, KSA

Michael Gebhart, Gebhart Quality Analysis (QA) 82, Germany

Marko Jäntti, University of Eastern Finland, Finland

ICSEA 2013

Forward

The Eighth International Conference on Software Engineering Advances (ICSEA 2013), held on October 27 - November 1, 2013 - Venice, Italy, continued a series of events covering a broad spectrum of software-related topics.

The conference covered fundamentals on designing, implementing, testing, validating and maintaining various kinds of software. The tracks treated the topics from theory to practice, in terms of methodologies, design, implementation, testing, use cases, tools, and lessons learnt. The conference topics covered classical and advanced methodologies, open source, agile software, as well as software deployment and software economics and education.

The conference had the following tracks:

- Advances in fundamentals for software development
- Advanced mechanisms for software development
- Advanced design tools for developing software
- Advanced facilities for accessing software
- Software performance
- Software security, privacy, safeness
- Advances in software testing
- Specialized software advanced applications
- Web Accessibility
- Open source software
- Agile software techniques
- Software deployment and maintenance
- Software engineering techniques, metrics, and formalisms
- Software economics, adoption, and education
- Business technology
- Improving research productivity

Similar to the previous edition, this event continued to be very competitive in its selection process and very well perceived by the international software engineering community. As such, it is attracting excellent contributions and active participation from all over the world. We were very pleased to receive a large amount of top quality contributions.

We take here the opportunity to warmly thank all the members of the ICSEA 2013 technical program committee as well as the numerous reviewers. The creation of such a broad and high quality conference program would not have been possible without their involvement. We also kindly thank all the authors that dedicated much of their time and efforts to contribute to the ICSEA 2013. We truly believe that thanks to all these efforts, the final conference program consists of top quality contributions.

This event could also not have been a reality without the support of many individuals, organizations and sponsors. We also gratefully thank the members of the ICSEA 2013 organizing committee for their help in handling the logistics and for their work that is making this professional meeting a success.

We hope the ICSEA 2013 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in software engineering research. We also hope the attendees enjoyed the charm of Venice

ICSEA 2013 Chairs

ICSEA Advisory Chairs

Herwig Mannaert, University of Antwerp, Belgium
Jon G. Hall, The Open University - Milton Keynes, UK
Mira Kajko-Mattsson, Stockholm University & Royal Institute of Technology, Sweden
Luigi Lavazza, Università dell'Insubria - Varese, Italy
Roy Oberhauser, Aalen University, Germany
Elena Troubitsyna, Åbo Akademi University, Finland
Luis Fernandez-Sanz, Universidad de Alcala, Spain
Michael Gebhart, Gebhart Quality Analysis (QA) 82, Germany

ICSEA 2013 Research Institute Liaison Chairs

Oleksandr Panchenko, Hasso Plattner Institute for Software Systems Engineering - Potsdam, Germany
Teemu Kanstrén, VTT Technical Research Centre of Finland - Oulu, Finland
Osamu Takaki, Japan Advanced Institute of Science and Technology (JAIST) – Ishikawa, Japan
Georg Buchgeher, Software Competence Center Hagenberg GmbH, Austria

ICSEA 2013 Industry/Research Chairs

Herman Hartmann, University of Groningen, The Netherlands
Hongyu Pei Breivold, ABB Corporate Research, Sweden

ICSEA 2013 Special Area Chairs

Formal Methods
Paul J. Gibson, Telecom & Management SudParis, France

Business and process techniques
Maribel Yasmina Santos, University of Minho, Portugal

Testing and Validation
Florian Barth, University of Mannheim, Germany

Web Accessibility
Adriana Martin, National University of Austral Patagonia (UNPA), Argentina

ICSEA 2013

Committee

ICSEA 2013 Technical Program Committee

Table of Contents

Kongdroid: A proposal for a Cloud Service for Stress Testing on Android Applications

Leonardo M. A. Sodré, Felipe Silva Ferraz, Gustavo Henrique da Silva Alexandre, Ana C. L. De Carvalho

CESAR
Centro de Estudos e Sistemas Avançados do Recife
Recife, Brazil
{lmas, fsf, ghsa, alcl}@cesar.org.br

Informatics Center
Federal University of Pernambuco
Recife, Brazil
{fsf3, ghsa}@cin.ufpe.br

***Abstract* – This work proposes a new and scalable service for stress testing on Android applications. This tool is available through cloud computing resources to support developers in their applications validation, aiming robustness, stability and compatibility, in different devices before commercial deployment. The solution focuses on the generation of a certain number of pseudo-random user interface events in the installed application in an emulator. This emulator is created from real images, of customized versions of the Android platform, running in well known devices. This execution results in a report containing the events that were successfully and those that failed due to any specific reason.**

Keywords-cloud computing; stress testing; remote testing; mobile applications; Android

I. INTRODUCTION

The software development, for mobile devices, and the conduction of large-scale experimental developing studies using real person, have become easier through the creation of app stores, and by using those stores as a mechanism for a significant number of users, to publish applications they have authored. An example of this was the emergence of the Apple store, which popularized this type of service. Unlike Apple's iOS platform, Google's Android open platform does not impose restrictions on its operating system; thereby, creating favorable conditions for various hardware manufacturers to adopt these devices. However, this benefit comes at a price: the challenge has become to develop interactive applications that need to run on a variety of these manufacturers' items of hardware equipment, each with its own customized version of the operating system, different hardware resource capabilities and screen resolutions and functionalities. Another relevant factor is the evolution of the Android version, where the application needs to track changes on the platform to keep operating properly.

Taking advantage of this benefit of the open platform and manufacturers' mass launch of more affordable Android devices, according to a recent survey, last year, the Google Store tripled in size, with its stock in 2013 amounting to about 800 (eight hundred) thousand applications [1], and recorded more than 25 (twenty five) billion downloads in 2012 [2].

Even though this demand has created the benefit of a proliferation of applications, it has also presented the need to address a growing issue: they have difficulty in generating various user events to stress the application and check if any exception occurs; in testing the capacities and resolutions of Android devices on different models; and there are few physical models available for testing. This difficulty of having an insufficient number of devices is also a reality faced by organizations.

Among the techniques used in this study was that of using an Android emulator instead of the physical model. This is because unlike the iOS simulator [3] and its resource constraints, the emulator reproduces a real device efficiently. This decision to use an emulator was further strengthened when it became feasible to configure the emulator, released by the manufacturer, with real versions of the Android platform. Given the support of cloud computing resources, it was possible to pre-configure these emulators in a scalable environment, thus enabling it to be used in parallel, so as to meet users' requirements as to running their application on several mobile devices. By means of an Application Programming Interface (API) accessed through an Internet browser, the user accesses this cloud environment to subject his/her application to testing, for which a script will be generated automatically to install the app in the emulator and apply the stress command using the Android Monkey tool, native to the platform. If the processing of the test demands a high consumption of infrastructure resources so as not to compromise the run quality, a new instance may be used to balance these resources in order to ensure the delivery of the results.

The program put forward in this paper to tackle these difficulties is called a Kongdroid. This enables the developer to use a prepared and configured environment in which to conduct stress testing [4]. It is hoped that, by having this facility, the knowledge of test development that a developer needs will be reduced and that time will be gained as there is no need to prepare an infrastructure since this is provided by this service. As a result of using the Kongdroid, it is estimated and it will permit the publication of more robust applications that are compatible with various Android device models, i.e., that it will indicate possible areas for improvement, so as to anticipate corrections, while the model is still in the development phase. Problems of the type in which the application is unexpectedly closed are among the situations that are not so easy to spot, unless features such as the Android Monkey [5] are used.

This was a structured study, which began with the authors deepening their knowledge of the technologies used and a review of the literature so as to be able to cite related studies. This introductory section draws attention to the state of the art with the issues related to Kongdroid. Then the proposed solution is detailed by describing the techniques used to create the service and matters to be careful about and points to consider. These strongly guided the study while it was being developed. After recording the approach to finding a solution, an account is given of the planning, implementation and the comparison of two experiments undertaken in which the solution was applied so as to give evidence of how important it is to use it. To summarize all the work done, the concluding section indicates the improvements achieved in the state of the art, the advantages and limitations of Kongdroid, its possible applications and ideas on how it may evolve.

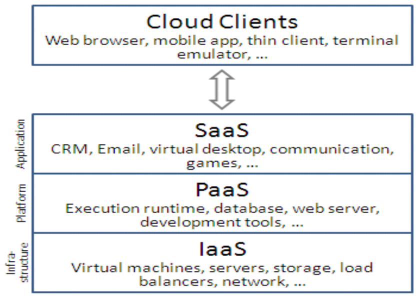
This paper is divided as follows: The first section presents a short introduction, section II presents the state of art about topics used in this paper, section III presents the proposed solution and how it was developed, the experiments and results used to validate the tool are depicted in section IV, finally section V presents some conclusions about this work.

II. State of the Art

A. *Cloud Computing*

Cloud Computing [6] is the representation of the applications made available as a service on the Internet and by *software* and *hardware* in the data centers that make these services feasible [7]. There are many definitions of Cloud Computing, but some features are held in common by most of them, for example, virtualized environments and providing computing resources on demand. This type of service is commonly called a public cloud. A private cloud is a center with data restricted to a specific company or of limited access [8].

Cloud Computing is divided into three main types to offer services, as shown in Figure 1: Software as a Service (SaaS) [9], Platform as a Service (PaaS) [10], and Infrastructure as a Service (IaaS) [11].


Cloud Clients
Web browser, mobile app, thin client, terminal emulator, ...
SaaS
CRM, Email, virtual desktop, communication, games, ...
PaaS
Execution runtime, database, web server, development tools, ...
IaaS
Virtual machines, servers, storage, load balancers, network, ...
Application
Platform
Infra-structure
Figure 1. Cloud computing at different levels

Related to this work, there is a type of cloud computing called Testing as a service (TaaS), which offers users testing services, such as the automatic generation of test cases, automated conduct of tests and evaluation of test results [12]. Testing tasks can be modeled using ontology techniques, and they can be combined based on a shared ontology model, along side with TaaS, there are, other subtypes: Development as a Service (DaaS) [13], Communications as a Service (CaaS) [14] and Everything as a Service (EaaS), which are not part of this scope.

B. *Android Platform*

Android [15,16] is a platform for mobile devices that runs on the nucleus of the Linux operating system but developed into a structure external to this nucleus [17]. The Android operating system was initially developed by Google and later by the Open Handset Alliance (OHA), which is a group of large companies in the telephone mobile market such as HTC, LG, Motorola, Samsung, Sony Ericsson, Toshiba, Nextel, China Mobile, T-Mobile, ASUS, Intel, Garmin and others. OHA is led by Google and the group's goal is to define a single open platform for mobile phones; thus, making consumers more satisfied with the final product. Another goal of the group is to create a flexible platform on which to develop applications. The birth of Android came about based on these objectives for which OHA is responsible for maintaining a standard platform where all the new market trends are present in a single solution [17, 18].

Android applications in [19] are built using Java language; but, there is no Java virtual machine in the operating system, only a virtual machine optimized for mobile devices called Dalvik [20, 21].

C. *Monkey Test*

Android Software Development Kit (SDK) [22] makes a Monkey test tool available to generate pseudo-random user events such as clicks, touches, or gestures and other events at the system level. As the guide to the Android platform itself says, "You can use the Monkey to stress-test applications that you are developing, in a random yet repeatable manner "[5].

The Monkey is a tool accessed via the command line that can be run on an instance of the emulator or mobile device. There are four main categories of options: basic configuration, such as the definition of the number of attempts for random events; operational restrictions, such as restricting the test to a single package; event types and frequencies; and debugging options [5]. During these events the tool observes three conditions, which deal specifically with the following: if it is restricted to execution in one or more specific packages, it watches for attempts to browse for other packages, and blocks them; if the application crashes or receives any type of *not-dealt-with* exception, the Monkey will stop and report the error, and if the application generates an *application not responding* error, the Monkey will stop and report the error [5]. Other types of behaviors of defects that the Monkey does not detect can be mapped by other types of smart Monkey tools [23]. Other related studies use stress testing: AASanbox [24, 25] and model-based GUI for Android applications [26].

D. Testdroid

The Testdroid is a useful tool for Android application developers who can validate if their application is compatible with several other types of devices [27, 28]. It is proposed to perform a specific set of user actions on one or more real device and collect and report test results. It is a service that is available on the Internet, for which the steps: Record your test, Run test on real devices and check reports.

III. Proposed Solution

The proposed service is committed to providing a check on the user's application, using stress testing [4], based on the native Monkey Test tool of the Android platform, to validate the robustness and compatibility in various telephone options and other mobile devices. After it has been run, reports of the results are generated for data analysis and emailed to the client. With such data, the client will obtain valuable information to support improving the application and ensuring quality, as shown in the proposed high-level architecture in Figure 2. It will also lead to a better understanding of the flow of the run and the entities involved. In the following sections, this paper will describe this solution in greater detail so as to understand its methodology, structure and development decisions. Real devices are dispensed with because the tests are run exclusively on emulators.

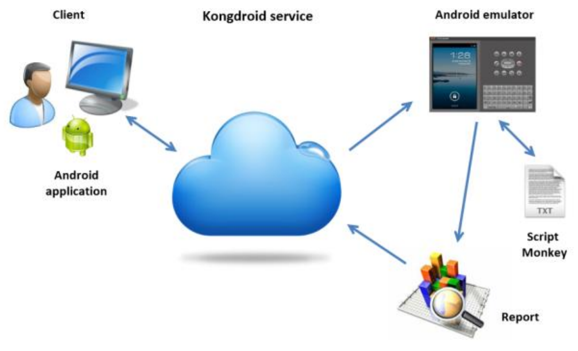


Figure 2. High-level definition of the architecture proposed

A. Definition of the architecture based on the cloud

After a detailed study of the necessary functionalities of the solution, it became very clear that to meet the user demand, the architecture should have the following quality attributes:

- Availability: The system will be available 7 days a week and 24 hours a day;
- Integrity/security [29, 30]: Only users with access privileges may configure and run tests. Every application transferred to the service and tested will be discarded at the end;
- Interoperability: The solution should be able to operationalize its being implemented in different modules, management and others, to conduct testing processes, running on different operating systems, Windows and Linux, respectively;
- Usability: A new user should be able to conduct a test of an application without the need for guidance, only with the support of tips on the filter options of the commands;
- Scalability [31]: The service should scale computing resources whenever there is a need to ensure the correct balancing of the processing of users' requests.
- Use of standards [32, 33]: The solution should support pre-established script models for running tests, so that they are dynamically created from the selection of the options of mobile devices.

To meet these requirements, an infrastructure benchmark on the market was adopted and widely used by several companies, *Amazon Web Services* (AWS) [34]. AWS offers a variety of cloud services, of which the one that stands out is Amazon Elastic Computer Cloud (EC2) [35], which permits the rental of instances of virtual servers that can be scalable to the extent that the solution needs both processing and to place limitations on software.

B. Definition of the standards used

In order to structure a better service, it was very important to define models and nomenclature standards and the target of the resources. In the presentation of the solution to the user, there is the possibility of selecting more than one type of mobile device. This feature led to a considerable complexity, since the architecture should be flexible enough to allow the addition of new devices without causing the work developed to be reworked. In meeting this functionality, for each model made available, a base script model to carry out the commands of the test is defined. After the executive action of the user, this standard model is used, based on the filters selected by the user, to generate dynamically the final script for conducting the test.

1) Nomenclature of commands and resources

First of all, the target folder for each authenticated user was identified, named by his/her identification in the system. In this folder, what are stored are all the resources to be used such as: application to be tested; test script commands for mobile and log files of the relevant events, commonly called a log.

A unique identifier is assigned to each mobile device option. Thus, the identifier is used for all command scripts generated. For the identifier "001", the script will have to be generated with the following format: Script_Monkey_001.bat. It is also used to generate the logcat (relevant phone events) with the following format: LogCat_001.log and generation of the Monkey test log (relevant events of the stress test command) with the following format: LogMonkey_001.txt.

One of the commands carried out by the command script is the startup of an Android *Virtual Device* (AVD) to start the emulator. For each type of mobile device, an AVD was created with the following format: avd [identifier], ie, for the device with the ID "001", the nomenclature is avd001.

C. Preparation of the environment

As cited in the definition of the architecture, the AWS infrastructure was chosen to make the service available in the cloud. For this setting to work properly, an extensive level of knowledge of managing servers or operating system processes was not required, but some settings are essential so it works correctly.

1) EC2 Structure

For this study, a new account was created on the AWS and the EC2 service used to create the instance of the "t1.micro" type on the platform of the Windows operating system, Server 2008 R2. This type of instance has limited resources (CPU and RAM memory) and its use is free for one year, i.e., payment for its use is not required unless use exceeds some preset limits. When the instance is available, access can be gained through a *Domain Name System* (DNS) with a dynamic IP (internet protocol) address. This is not the best option because at every reboot of the instance, this address is modified. To overcome this drawback, the EC2 service has an elastic IP resource, i.e., for the public DNS, it is assigned a static IP address, thus ensuring there is always access to the same address.

2) Configuration of the Android platform

To use the Android emulator platform and to carry out the Monkey commands, the Android *Software Development Kit* (SDK), version 21, and the Java *Runtime Environment* (JRE), version 1.7 have to be installed in the AWS. Environment variables were created: "ANDROID_SDK_HOME" containing the path of the Android SDK and "JAVA_HOME", containing the path to the JRE.

After properly installing the Android, the Android SDK *Manager* had to be run to complete the upgrades of associated tools. Among these updates, one requires special attention, namely, the Google APIs Add-On. The add-on provides system images compatible with Android that runs on the Android emulator, thus enabling the application to be debugged, run and tested before publishing it to users. Several mobile phone manufacturers have these images on their web pages targeted on application developers. For this study, the images used were from the Motorola manufacturers: Atrix 2 and Razr and LG 3D Optimus model [36,37].

3) Configuration of the Microsoft platform

To implement the solution developed in ASP.NET MVC 4 [38], it was necessary to install the *Internet Information Service* (IIS) version 7.5 and the Microsoft. NET Framework 4.5 in the AWS. For IIS, it was necessary to create an application called "monkey", where the implementation of the solution was stored and the right of full access to the folder called "Content" of the application was assigned to the user of the IIS (DefaultAppPool), so that "Content" allows the resources used to be stored and altered.

D. Model of Monkey script

As previously mentioned, to make the service flexible as to replacing and/or adding new options for mobile devices, a script model was created to run the commands needed to perform the stress test. This stage of the project required close attention and simulations to determine the optimal sequence of actions to ensure better efficiency in the results hoped for. The use of Android emulators involves a series of difficulties when they are in an automation process, since the ability to foresee the time needed to trigger each command is not precise, and, therefore, auxiliary actions were used to minimize this uncertainty. Other resources were also taken advantage of to have the emulator perform better, since there was not the need for a graphical display. The automated commands in this script can be run manually in the user's environment, but they involve complexity in configuring the necessary tools and environment variables.

The identifier of each mobile device option was parameterized in this model so that all resources accessed and generated are easily referenced, based on the data selected by the user, the script is easily generated and applied in the environment of the solution.

1) Selection of port

For this automation would function properly, the environment was totally controlled, i.e., for each script generated a known number of the network port is generated and later will be attributed to the Android emulator. This strategy is of fundamental importance to free the memory of the emulator at the end of the test. The generation of the port number is made at random between 5554 and 5584, this range being reserved for this type of program. By default, if the port does not specify it, it is associated with the generating the numbers 5554 and 5555 (this second port is reserved for the *android debug bridge* (ADB)) and should another emulator be run in parallel, the next port is that of the number 5556, and so on, successively.

The following is an example of a command linking port 5558 to the emulator:

```
emulator -ports 5558,5559 -avd avd001
```

In the example below, there is a sample command for a specific emulator, for the installation of the user's APK on the emulator:

```
adb -s emulator-5558 install helloWorld.apk
```

2) Estimate of time for each step

When using the mobile device or emulator, the focus of this study, in a process of stress test automation [39], the time it is estimated time for each command to be executed should be taken into account. If there is not enough time left over for the next command to be applied under favorable conditions, the procedure, as a whole, will be compromised and aborted. To ensure the efficiency of the script actions, possible points of delay were identified and auxiliary commands were defined in order to be used following these main ones, i.e., promoting a longer time so that the environment is in a fit state for the next step. One difficulty found was that the operating system does not provide a specific command for this situation, where, to solve this limitation, another command was used to obtain the same result. The following is an example:

```
ping 1.1.1.1 -n 1 -w %_timer% >NUL
```

For an operating system with a TCP/IP client, the PING command can be used to delay the run by a number of seconds. If specified (-w), the PING will wait for a number of milliseconds between two pings before giving a time limit. The environment variable, represented by (*% timer%)*, contains the time that the action will imply.

This feature was used to overcome three points of slowness:

- Running the Emulator: Estimated time of 240,000 (two hundred forty thousands) milliseconds;
- Installing the Android Application Package (APK): Estimated time of 20,000 (twenty thousand) milliseconds;

Conduct of the Monkey test: Estimated time of 120,000 (one hundred and twenty thousand) milliseconds. A fixed value was used due to the project being limited to 500 (five hundred) random events. In an environment with a high processing infrastructure, without limitation on events, this estimate would need to use a formula such that the time might vary proportionally.

3) Tool for recovering the APK package

In order for the command for the stress test to be able to restrict the target application, it is fundamental to know the name of the package that will be used as a parameter. For the purposes of promoting a better experience for the user, when using the service to enter and select data to perform the test, there is no need to register this package in order to avoid errors when typing manually.

To meet this situation, a tool called *android-apktool* was used. This is a tool available in the repository of Google projects under the *Apache License 2.0*, which undertakes reverse engineering on Android APK files. It can decode resources to nearly their original form and rebuild them after some modifications have been made. Thus it was possible, starting with the APK user, to decode the information of the package and use it as a parameter in the command of the stress test.

4) Definition of variables

When defining the script model definition, some temporary environment variables were created to make it possible when the script was generated to have a specific one for the mobile device model and for the dynamic use of information in the commands to be executed.

The example below better illustrates the need to use these variables:

```
%_adbPath% -s %_serialEmulator% install
%_apkPath%
```

The same command used in session 3.5.1 to install the user's APK, but this time the variable *%_adbPath%* was used, which identifies the path of the Android ADB program to carry out the commands, *%_serialEmulator%*, which identifies the serial or port in which the emulator is running, and *%_apkPath%*, which identifies the path of the user's APK stored on the server.

5) Command to optimize the emulator

To avoid overloading the server, should more than one instance of the emulator be run, unnecessary features in an environment may be discarded without interacting with the user. Thus, the options of initial animation, graphical and audio interface were disregarded. The following is an example of the command:

```
emulator -ports 5554,5555 -no-boot-anim -no-window -
noaudio -avd avd001
```

6) Command to instal the Apk

After the above command to run the emulator, the next to be auctioned is to install the user's APK. The ADB provides an option so that this action occurs only when the emulator is "ready", thus avoiding error and the script being interrupted. The following is an example of the command used:

```
adb -s emulator-5558 -e wait-for-device install -r
helloWorld.apk
```

7) Command to unblock the screen

During the period of testing the solution it was realized that the emulator on being started, by default, is left with its screen blocked. Thus the command of the stress test was discarded. To resolve this issue, a command was included in the script to send a screen unblock event. Below is an example of the command used:

```
adb -s emulator-5558 shell input keyevent 82
```

8) Command to conduct the Monkey

For the main action of the script, the following command to run the stress test was specified:

```
%_adbPath% -s %_serialEmulator% shell monkey -v -v
-p %_apkPackage% " + monkeyParameters + "
%_monkeyEvents% > %_logMonkeyPath%
```

The arguments-*v-v* promote greater information in the tests run. The *%_apkPackage%* variable stores the name of the package, extracted as described in section 3.5.3. The *monkeyParameters* development variable stores the set of options for user-selected parameters. The *%_monkeyEvents%* environment variable stores the number of interface pseudo events reported by the user. The *%_logMonkeyPath%* variable stores the service path to record the results of the stress test.

E. *Conduct of the test*

To run the tests it was defined that the service should possess a modularized and simplified flow. Using few steps, the application meets a demand and then has the capacity to quickly return to the initial state for a new request from the user.

As shown in Figure 3, after the user obtains his/her authentication, he/she is directed to the starting point of the service. The first piece of information requested is the submission of the application package to be tested, the Apk Android. This transfer is performed securely and at the end of the process it should be discarded. Still on the main screen the user will need to provide other important pieces of information, such as the email to, which results should be sent, to select which application of the device models should be validated, the number of pseudo events and other optional choices regarding the stress test, these being Monkey event options and Monkey debugging options. After completing the data and confirming the start of the operation, the system will validate them and if there is no criticism, the service will be started.

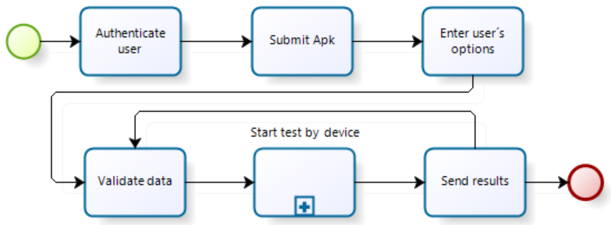


Figure 3. Main flow of the conduct of the test

From this point on, the system has already allocated the user´s physical space and the Apk Android is available for use in the emulator, such that the script should be generated and executed by the device model chosen. After each run of this script has been concluded, an email will be sent to the user and the result attached.

Figure 4 below shows each step of the test run by the model of the device selected. This process is performed in parallel so that the service does not take up the hardware resources of the AWS infrastructure for a lengthy period of time. Each run of an emulator requires a high level of processing and memory, in which the orchestration of these elements monitors the need to allocate more resources, i.e., whether another server will need to be initialized to balance and ensure the quality of the system.

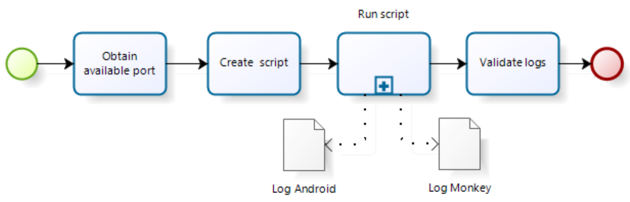


Figure 4. Secondary flow of the test per device

In the flow of Figure 4, the first step is to check and select the port number of the server where the emulator will be allocated. This port is one of the parameters used for the next step, the creation of the script. At this point, what are defined are the times between each execution of a command are defined, the parameters entered by the user to compose the command Android Monkey command, the physical path in the server of the user's location to generate the results and the path for the Apk of the target application of the tests. During this run, a log file of the events generated from the emulator and another log file of the events of the stress test are generated with the test result. The flow is finalized with the validation of these files.

Figure 5 below shows the sequence of the commands that make up the test script. The run starts by using environment variables used during actions in the emulator. Via the *android-apktool* tool, the name of the Apk package is recovered and stored in an environmental variable to be used later in the command of the stress test. The next step is to run the emulator, in which, to ensure optimum performance, parameters are used to bypass the startup animation, the audio and screen. At this point, the generation of events of the emulator is also triggered. The next action is to install the Apk of the application which, via the *wait-for-device* parameter, is only run after the emulator is found in the *device* state, i.e., its instance is prepared to respond to the user's actions. After the Apk has been installed, but before running the stress test command, the screen must be unblocked, since without this step the pseudo events of the Monkey android are prevented from interacting with the application. From this point on, the emulator has the necessary condition for the stress test to start and to record on file the events in order to compose the result, whether there was a failure or success.

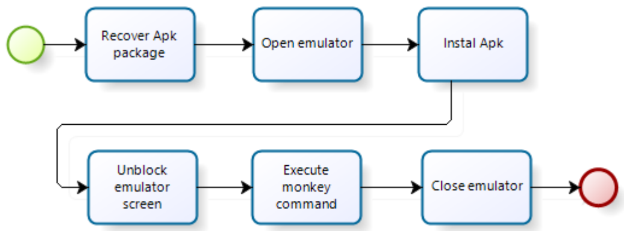


Figure 5. Secondary flow of the execution of the test script

After all the commands have been carried out, the Apk application is uninstalled and deleted and the instance of the emulator is closed.

F. *Summary of the solution*

Figure 6 illustrates the architecture of the solution at a more detailed level, where the user, via a web browser, submits his/her application and informs the proposed cloud service of the parameters desired, which are loaded to run and scale in an orchestrated way all the resources required, such as to ensure the expected results.

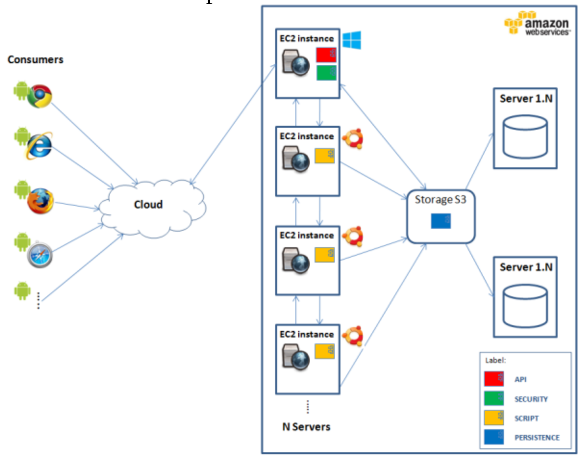


Figure 6. Low-level definition of the architecture proposed

As Figure 6 shows, the first instance used of a virtual server is that of a Windows Server, which is responsible for starting the service and using the data selected by the user, it dynamically generates scripts, per device, to be run. In the second moment, another instance of a virtual server is initialized, but this time is used for the option of a Linux Ubuntu machine. The scripts of the stress test are run in parallel in this new instance; should it be necessary, another instance with the same settings can be used without compromising the total flow of the solution. After finalizing the conduct of the stress test, a report is stored and sent to the user so he/she can analyze it.

IV. Experiments and Results

To prove the correct functioning of the entire solution, two examples run on Kongdroid will be described. The input parameters and the expected result will be specified, as well as a comparative analysis to prove why using the tool as a support tool for developers of applications is important before publication to future users.

A. *Experiment undertaken*

In the selection of the applications, the following strategy was used: both should appear as published in the Google Store (Google Play), an example of a simpler application with a satisfactory result, and another example of an application of more moderate complexity with a fault in the test of the application not responding (ANR) type.

For the simple application, one was selected from the calculator type, categorized as a utility, called *Shake Calc*. It is proposed to be a scientific calculator with the following features: accelerometer to finalize the calculation, basic vision for access to the more frequently used functions and more complex calculations; it can switch to an advanced mode of exhibition with a touch from the user, as shown in Figure 8. For the application of moderate complexity, one was selected of the type with tables, categorized as children's games, called *Smart Bubbles.* Figure 7 shows the mentioned game that is proposed to be a math table with the following features: a game to learn the tables in a fun way, during the game, equations and bubbles with numbers are presented; for each equation, a bubble appears with the correct result and some others with wrong results.

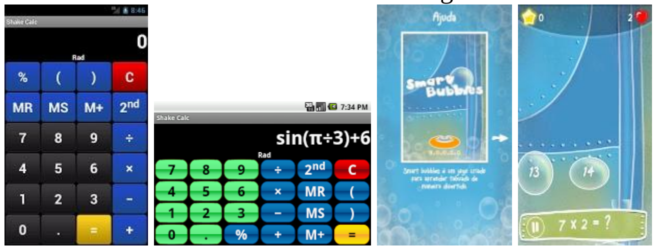


Figure 7. The first two figures represent Shake Calc and the last two Smart Bubbles

The Kongdroid was started after being informed of the following input parameters: Choice of the APK of the application to send to the service, the email to receive the results, the target device of the stress test selected (for the *Shake Calc*, the Motorola Atrix 2 model was used and for the *Smart Bubbles*,the Motorola Razr was used). It was also informed of 500 (five hundred) pseudo random events, 300 millisecond gaps between each event, to ignore *crashes*, to ignore *timeouts* and to ignore *security exceptions*. These latter three parameters are generally used to provide for the test being run completely, unless it is finalized by the operating system.

B. *Metrics used*

To better measure and condition the comparison of the results for a more realistic analysis, the metrics were defined of the total number of events per the number of events run and the number of application runs by the number of applications successfully tested.

C. *Results*

After submitting the APK application to Kongdroid and informing it of the input parameters for each application of the experiment, the cloud service will process the stress test. This step takes less than ten (10) minutes, since every action has a maximum time configured to be run on the application installed on the Android emulator for greater efficiency in allocating and releasing resources, as well as in the response time of the results to the user. Upon completion of the due tests, an email is created with the log files of the environment and the Monkey with a text attached, whether the test was successful or not, and sent to the user´s email address so he /she may investigate and analyze the results.

In the most significant part of the Monkey log file of the stress test done on the *Shake Calc* application, it is observed that the test was successfully completed by the text "// Monkey finished", where all five hundred pseudo random events were run without an exception having occurred.

In the most significant part of the Monkey log file of the stress test done on the Smart Bubbles application, as cited when planning the experiment, the log highlights the failure of the test by the ANR type of error that occurred, where the application on receiving a given pseudo random user event, a certain time without a response which leads the operating system to cause the error in the application and to close it immediately so as not to compromise other functionalities of the device. If another type of error occurred in the application, it would also be recorded on the Monkey log.

D. *Comparison of the results*

Metrics were applied with the following evidence:

- Number of events per total number of events run: For the *Shake Calc* application of the five hundred pseudo random events programmed all were successfully run. For the *Smart Bubbles* application of the five hundred scheduled events only twenty-five were run successfully. As shown in the graph in Figure 8:

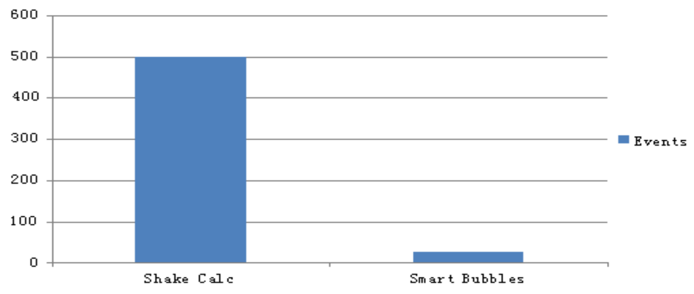


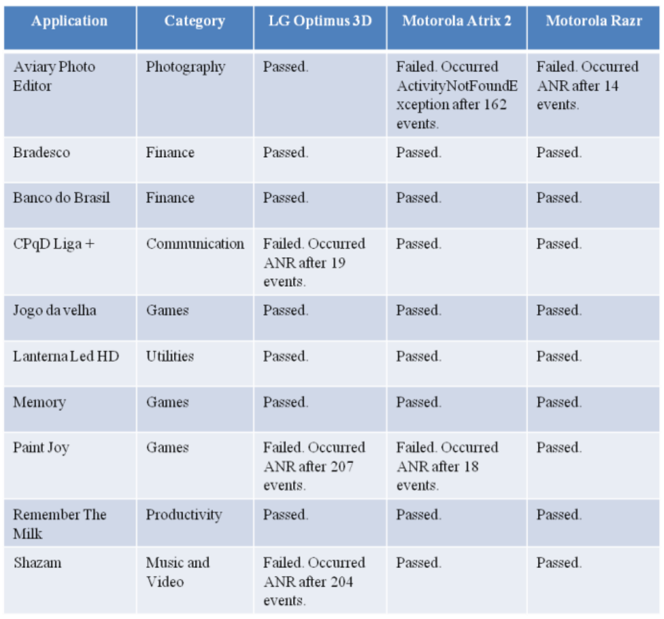
Figure 8. Number of random pseudo events

- Number of applications per number of applications successfully tested: Two applications selected for the experiment, where one had a successful test (*Shake Calc*), and one had a failure in the test (*Smart Bubbles*).

By using the results of the metrics, very different scenarios and conclusions can be obtained. While it was attested that the experiment conducted with the *Shake Calc* application, after subjecting it to a significant load of user events, its stability responded effectively, thus ensuring that its publication and other devices running on Android had greater reliability, in the experiment conducted with the *Smart Bubbles* application, it was proven that it does not have the efficiency to withstand a greater number of User Interface events, in which when a severe ANR error occurs, the application needed to be finalized by the operating system.

This type of error could be avoided in the development phase by using a tool like Kongdroid, so that the credibility of the application is not threatened. This is a real threat given that the application is published and the user on downloading it could come across the kind of situation where he/she may suddenly be impeded from continuing to use it and which may easily cause that the application can no longer be used.

To better attest the efficiency of this work, another ten (10) applications from the Google Play store were selected, all of which were downloaded by a significant number of users. The tests were performed on three device models offered by Kongdroid, LG Optimos 3D, Motorola Atrix 2 and Motorola Razr. The following Figure 9 shows the results of the stress tests:



Application	Category	LG Optimus 3D	Motorola Atrix 2	Motorola Razr
Aviary Photo Editor	Photography	Passed.	Failed. Occurred ActivityNotFoundException after 162 events.	Failed. Occurred ANR after 14 events.
Bradesco	Finance	Passed.	Passed.	Passed.
Banco do Brasil	Finance	Passed.	Passed.	Passed.
CPqD Liga +	Communication	Failed. Occurred ANR after 19 events.	Passed.	Passed.
Jogo da velha	Games	Passed.	Passed.	Passed.
Lanterna Led HD	Utilities	Passed.	Passed.	Passed.
Memory	Games	Passed.	Passed.	Passed.
Paint Joy	Games	Failed. Occurred ANR after 207 events.	Failed. Occurred ANR after 18 events.	Passed.
Remember The Milk	Productivity	Passed.	Passed.	Passed.
Shazam	Music and Video	Failed. Occurred ANR after 204 events.	Passed.	Passed.

Figure 9. Results in another 10 applications

Among the related studies presented in this paper and other test automation tools surveyed, characteristics similar to those in Kongdroid were not found. Due to this, it was difficult making it possible to compile a valid comparison test to attest to its efficiency. This is why the focus of the experiments and results was on validating the quality of existing applications in the Google Play store when subjected to stress tests in different mobile device models.

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented the Kongdroid, a cloud computing service to automate stress testing so as to analyze Android applications. It is shown how the Android emulator can be used to run applications in an isolated and pre-configured environment with real images of versions of operating systems released by device manufacturers. The main purpose of this solution is to enable developers to subject their application to a high number of pseudo random user events in various Android devices to assure their effectiveness as to the correct conduct of the functionalities.

Its importance is due to the fact that of its offering a thorough knowledge of stress testing techniques, where the developer will be able to use a pre-prepared environment to validate his/her application in various device models with different capacities and resolutions.

The advantages of using this service are obtained because of the detailed results of the environment and events performed being sent more speedily to the user for his/her analysis. This makes it an important tool in supporting development in order to pinpoint quickly areas to be improved before publication in the Apps store. The previous limitation that the developer had due to restricted use for testing on devices no longer exists.

Among the limitations of the service, there is the difficulty of repeating the test effectively, restricting the stress test to one application screen, the difficult of closing a specific instance of the Android emulator in the Windows environment, the absence of images of the Android platform for a given mobile device model and the high consumption of memory and the limit of instances of the emulator.

The results obtained from the experiments undertaken show there is no effective control by the Google Store as to effective compatibility of their applications in the different models found in the market. In this case, the assurance needs to come from the very author of the application using a tool such as Kongdroid.

One of the main contributions of this paper was that of permitting the developer the facility of lessening his/her need to acquire extensive knowledge of test development. Without requiring complexity when preparing a test environment, the service offers simplicity when generating a considerable number of the user´s interface events in the target application. This initiative enables the publication of the application to be more robust and compatible with various models of Android devices. Another important point is to anticipate improvements and corrections during the development phase, because what are avoided are problems of the type in which the application is ended unexpectedly during use. Besides costing less to correct before publication, this does not adversely affect the credibility of the author of the application.

For future research studies, we plan: adding new options for device models; a new mechanism for freeing the emulator at the end of the test for the Windows environment; a real-time listing of events being run; a test result in a more professional format; comparative results between devices tested; improving the performance of the emulator and; creating an orchestrator to manage cloud computing resources more efficiently

REFERENCES

[1] I. Paul. http://www.rssphone.com/google-play-store-800000-apps-and-overtake-apple-appstore/. Accessed in February 2013.

[2] Z. Lutz. http://www.engadget.com/2012/09/26/google-play-hits-25-billion-app-downloads/. Accessed in November 2012.

[3] M. Goadrich and M. Rogers. Smart smartphone development: iOS versus Android. In SIGCSE, volume 42, 2011.

[4] Ham K.H., Park, Y.B. 2011. Mobile Application Compatibility Test System Design for Android Fragmentation. CCIS 257, pp. 314-320.

[5] UI/Application Exerciser Monkey, http://developer.android.com/guide/developing/tools/monkey.html. Accessed in February 2013.

[6] Parkhill, D. The Challenge of the Computer Utility.

[7] A. Bechtolsheim. Cloud Computing and Cloud Networking. talk at UC Berkeley, December 2008.

[8] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. UC Berkeley, February, 2009.

[9] Candan, K. S., Li, W.-S., Phan, T., and Zhou, M. Frontiers in Information and Software as Services. In Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE2009), pp. 1761-1768, 2009.

[10] D. Cheng. PaaS-onomics: A CIO's Guide to using Platform-as-a-Service to Lower Costs of Application Initiatives while improving the Business Value of IT. Technical Report, LongJump, 2008.

[11] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (IAAS)", International Journal of engineering and information Technology, vol. 2, no. 1, 2010, pp.60-63.

[12] L. Yu, S. Su, J. Zhao, et al, "Performing Unit Testing Based on Testing as a Service (TaaS) Approach", Proceedings of International Conference on Service Science (ICSS) 2008, pp. 127-131.

[13] K. Matsumoto, S. Kibe, , M. Uehara, and H. Mori. "Design of Development as a Service in the Cloud" Network-Based Information Systems (NBiS), 15th International Conference on, (2012). Kawagoe, Japan 2012.

[14] L. Youseff, M. Butrico, and D. Da Silva. Toward a unified ontology of cloud computing. In Grid Computing Environments Workshop, 2008. GCE'08.

[15] S. Brahler, Analysis of Android architecture. Karlsruher Institut für Technologie, http://os.ibds.kit.edu/downloads/sa_2010_braehler-stefan_android architecture.pdf, accessed Nov 14, 2010, Outubro 2010.

[16] Android, www.android.com. Accessed in February, 2013.

[17] Google Android, Ricardo R. Lecheta, 2a Edição, Novatec, Junho/2010.

[18] Professional Android Application Development, Reto Meier, Wiley Publishing, Inc., 2009.

[19] How many lines of code does it take to create the Android OS? http://www.gubatron.com/blog/2010/05/23/how-many-lines-of-code-does-it-take-to-create-the-android-os/. Accessed in February 2013.

[20] Dalvik, code.google.com/p/dalvik. Accessed in February 2013

[21] David Ehringer. The dalvik virtual machine architecture. Technical report, Google, March 2010.

[22] Android Application Development, Rick Rogers et al, O'Reilly, 2009.

[23] N. Nyman, "Using monkey test tools," Software Testing and Quality Engineering magazine, vol. 29, no. 2, pp. 18–21, 2000.

[24] A.-D. Schmidt, H.-G. Schmidt, L. Batyuk, J. H. Clausen, S. A. Camtepe, S. Albayrak, and C. Yildizli. Smartphone malware evolution revisited: Android next target? In Proceedings of the 4th IEEE International Conference on Malicious and Unwanted Software (Malware 2009), pp. 1–7. IEEE, 2009.

[25] T. Bläsing, L. Batyuk, and A. Schmidt. An Android Application Sandbox System for Suspicious Software Detection. In 5th International Conference on Malicious and Unwanted Software, Berlin, Germany, 2010.

[26] T. Takala, and M. Katara. Experiences of System-Level Model-Based GUI Testing of an Android Application. In Fourth IEEE International Conference on Software Testing, Verification and Validation, Finland, 2011.

[27] Kaasila, J. Ferreira, D. Kostakos, V & Ojala, T (2012). Testdroid: automated remote UI testing on Android. Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia – MUM '12: Art. 28.

[28] Robotium, 2010. It's like Selenium, but for Android. Retrieved on 19th January, 2012 from http://code.google.com/p/robotium/.

[29] T. Mendhe, P. Kamble and A. Thakre, "Survey on Security, Storage, and Networking of Cloud Computing", International Journal on Computer Science and Engineering (IJCSE), vol. 4, no. 11, (2012) November, ISSN : 0975-3397.

[30] R. Byyya, C. Yeo, S. Venugopal, J. Broberg, and I. Brandic. Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. Future Generation of Computer Systems. vol. 25. no. 6. pp. 599- 616. 2009.

[31] L. Jain and S. Bhardwaj, "Enterprise Cloud Computing: Key Considerations for Adoption" International Journal of Engineering and Information Technology Vol 2 , (2010). IJEIT 2010, 2(2), 113-117 ISSN 0976-0253 (Online).

[32] M. Fowler, UML Distilled. Addison-Wesley, 1997.

[33] A. Leff, and J. Rayfield. "Web-Application Development Using the Model-View-Controller Design Pattern," Proceedings of the 5th IEEE Enterprise Distributed Object Computing Conference, 2001, pp. 118-124.

[34] K. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, et al. Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud. In CloudCom, pp. 159–168. IEEE, 2010.

[35] W. Vogels. A Head in the Clouds—The Power of Infrastructure as a Service. In First workshop on Cloud Computing and in Applications (CCA '08), October 2008.

[36] Motorola Solutions Developer, developer.motorolasolutions.com. Accessed in October 2013.

[37] LG Developer, developer.lge.com. Accessed in October 2013.

[38] ASP.NET MVC 4, http://www.asp.net/mvc/mvc4. Accessed in February 2013.

[39] M. Grechanik, Q. Xie, and C. Fu, "Creating GUI testing tools using accessibility technologies," in Proc. IEEE International Conference on Software Testing, Verification, and Validation Workshops. Washington, DC, USA: IEEE Computer Society, 2009, pp. 243–250.

Functional Software Testing: A Systematic Mapping Study

Gilmar Ferreira Arantes
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: gilmar@inf.ufg.br

Plínio de Sá Leitão-Júnior
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: plinio@inf.ufg.br

Auri Marcelo Rizzo Vincenzi
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: auri@inf.ufg.br

Fábio Nogueira de Lucena
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: fabio@inf.ufg.br

***Abstract*—Software testing is part of a set of activities that ensure high quality software. It primarily aims at revealing defects that have been inserted into a software at various stages of its development. In functional testing, test requirements are derived from software specifications. This paper proposes a systematic map (SM). Its planning and execution were based on questions formulated to investigate functional criteria/techniques related to: i) assessment methods, which have an effect on cost and efficacy; and ii) application scenarios, which define the type of software in which they are used. Furthermore, we assess the strength of evidence and threats to SM validity.**

Keywords-software testing; testing techniques and criteria; functional testing; systematic mapping.

I. Introduction

Software testing is a knowledge area within the field of software engineering, which strives for quality and continually contributes to process and product improvement. The test's main objective is to reveal defects in the software so these may be solved prior to any damage. Ideally, the testing activity must be systematic, and the techniques used must balance cost reduction and increase the levels of defect detection, should any exist. Each technique has a set of test criteria, which may be used during the conception, selection, and evaluation of a test set.

Among the different types of testing techniques, functional testing has an important role for software quality improvement as it complements other methods. Thus, it is relevant to: (i) know how functional testing criteria are employed; (ii) identify weak and strong points; and (iii) describe scenarios in which they are used.

This paper's contributions are obtained through a systematic mapping study. According to Wohlin et al. [1], it follows the same processes and principles used in systematic literature reviews, although it has different criteria for quality assessment and inclusion/exclusion of studies. Due to its wider and more varied range, both the collected data and the literature review are mainly qualitative. The research questions avoid any tendencies; instead, they are more specific and often relate to empirical studies.

The systematic map aims at answering the following questions pertaining to functional software testing:

- **Primary research question**: Which comparisons have been made between test criteria?
- **Secondary research question**: What is the application scenario for each functional testing criterion?

The purpose of the primary research question is to find weak and strong points of functional testing criteria through comparisons made between them. Many aspects are observed, i.e., application costs and ability to detect defects. This question is considered primary because it: (i) provides information on the type of application and limitations; (ii) determines factors influencing efficiency and efficacy; and (ii) contributes to the proposal of other approaches to functional testing.

The secondary research question aims to identify the type of software in which functional criteria are used. It establishes criteria range and determines its application and restricted use in some areas.

The rest of our paper is thus organized: Section II presents the systematic mapping protocol and how it was conducted. Section III shows the results as they relate to our research questions. Section IV discusses the strength of evidence and threats to validity of the primary studies selected. Finally, Section V is made up of final considerations and research implications.

II. Mapping Planning

The systematic mapping protocol was planned according to the model presented by Biolchini et al. [2]. This section explores the main points of the elaborated plan.

A. *Scope of studies*

The protocol identified the scope of the studies by considering:

1) **Population** – Scientific publications on software testing;
2) **Intervention**: Functional testing criteria.
3) **Results**:
   a) Properties, characteristics and comparisons between functional testing criteria;
   b) Application context of each functional testing criterion.
4) **Application** – association among functional testing criteria to help detect defects; support for an effective use of each criterion, in isolation or as a set; assistance for the proposal of new functional criteria.

B. *Search strategy for selecting primary studies*

The strategy for searching and selecting primary studies was defined according to the research sources, keywords, language, and types of primary studies selected for mapping:

1) **Criteria for source selection** – Electronic indexing databases and internet search engines.
2) **Search methods** – Manually and web search engine.
3) **Source listing** – Conferences, journals and technical reports indexed by IEEExplore, ACM Digital Library and Google Scholar.
4) **Language of primary studies** – English, due to its widespread use in scientific writing.

C. *Pilot search execution*

A search string was defined for each indexed database considering the research questions, their respective quality and amplitude traits, as well as the search strategy for selecting primary studies.

D. *Criteria and procedure for selecting studies*

1) *Inclusion criteria*:
   a) **IC**$_1$ – Papers mentioning any features of a functional testing criterion;
   b) **IC**$_2$ – Papers comparing functional properties;
   c) **IC**$_3$ – Papers comparing properties of functional and structural testing criteria, as well as those of the random testing technique.
2) *Exclusion criteria*:
   a) **EC**$_1$ – Papers in which software testing is only mentioned and is not the main topic;
   b) **EC**$_2$ – Papers discussing software testing, but whose focus is not on functional or random testing techniques;
   c) **EC**$_3$ – Papers discussing functional testing criteria, which are not in any of the criteria groups previously defined for analysis;
   d) **EC**$_4$ – Papers discussing functional testing criteria, although its focus is not mentioned in any of the categories previously defined for analysis;
   e) **EC**$_5$ – Papers describing systematic procedures for test criteria assessment, frameworks, benchmarks for the comparison of testing methods, but which do not actually make any comparisons;
   f) **EC**$_6$ – Papers comparing test methods, which do not include functional testing;
   g) **EC**$_7$ – Papers discussing functional testing related to formal specifications;
   h) **EC**$_8$ – Papers focusing on theoretical analysis with no practical examples of the approach.

E. *Selection process of primary studies*

1) *Preliminary selection process* – Retrieved papers were analysed by reviewers, who were responsible for reading titles and abstracts. Once a paper was considered relevant by the reviewers, it would be fully read.
2) *Final selection process* – All papers selected were fully read by at least one reviewer, who then elaborated a document including abstracts, methodologies and testing methods mentioned in each paper, as well as other related concepts.
3) *Quality assessment of primary studies* – Researchers assessed the selected papers according to the quality criteria defined by Ali et al. [3].

F. *Final selection*

The final selection was carried out through four phases. Phase 1 refers to the primary studies retrieved from the electronic databases after the application of search strings. Phase 2 corresponds to the studies resulting from the preliminary selection process. Some studies were excluded because their titles and abstracts did not pertain to our research questions. Phase 3 refers to the studies obtained from the final selection process. Some studies were also excluded once they were fully read for the same reason stated above. In Phase 4, some studies were excluded for their low quality according to the quality criteria defined during the planning stage of the systematic map. In summary, a total of 27 primary studies were selected, of which:

- 14 are from the IEEE database;
- 7 are from the ACM database;
- 4 are from Google Scholar;
- and 2 are directly from Universidade Federal de Goiás (UFG).

Figure 1 shows a distribution of studies spanning from 1978 to 2011. This time span corresponds to the publishing year of the oldest study retrieved from the search string and the year the mapping ended, respectively. The graph shows that the highest number of publications on this subject occurred in 2006 (a total of 6). Furthermore, between 2008 and 2011 there were fewer studies, but a continued interest for research in this area.

G. *Digraph of internal citations*

To illustrate primary studies that refer to one or more studies from the selected set, we constructed a directed graph (digraph) to identify entry and exit points. Figure 1 shows a representation of the digraph.

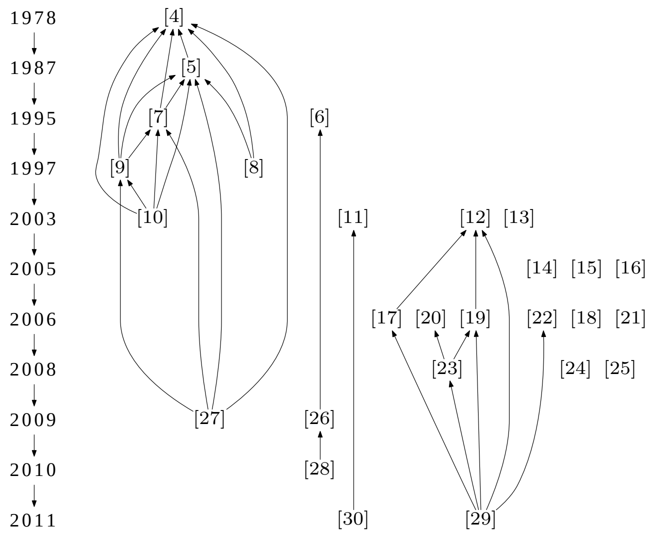


Figure 1. Citations among studies classified by year
.

Figure 1 reveals some areas of concentrated citations among primary studies. For instance, we identified an area of citations in which study [4] has the highest number of entries. This is due to the fact that it was one of the first published studies that approached the comparison of testing techniques. Another identified region includes study [29] with the highest number of exits. It is a survey, therefore it refers to many other primary studies. Finally, another region contains studies [6], [26] and [28], all of which use the same criteria for functional testing: Decision Table and Cause and Effect Graph.

III. Results

Table I presents testing criteria and techniques that were identified in the primary studies. The inspection approach is also used in these studies. The first column lists the criteria/techniques; in some cases, test approaches are not necessarily identified as a criterion, as stated in the literature. The second column shows the number of primary studies that use such criterion/technique. The third column lists the references used in the primary studies, and the last column indicates whether the criterion/technique is relevant to mapping. Thus, Table I shows that: (i) studies in general use more than one test criterion/technique; (ii) in many cases, functional, structural and other testing or code inspection techniques are compared in the same study; (iii) the following criteria are most used: Boundary Value Analysis, Equivalence Class Partitioning, and Decision Table.

Table I. Test criteria, techniques and approaches discussed in the studies analysed

Test Criteria/Techniques and Approaches	# Refs	References
Boundary Value Analysis	12	[5], [7], [8], [9], [10], [11], [13], [16], [18], [25], [27], [28]
Path Coverage	1	[27]
Statement Coverage	1	[5]
Condition Coverage	4	[7], [9], [10], [27]
Inspection/Code Review	6	[4], [5], [7], [9], [10], [27]
Cause-Effect Graph	3	[6], [26], [28]
Random Partitioning	1	[14]
Dynamic Partitioning	1	[14]
Equivalence Class Partitioning	11	[5], [7], [8], [9], [10], [11], [15], [16], [18], [27], [28]
Decision Table	6	[6], [15], [24], [26], [27], [28]
Test using Collaboration Diagram	1	[21]
Test using Object-Z	1	[21]
Test using OCL	1	[21]
Random Testing	2	[8], [14]
Use Case Test	6	[12], [19], [20], [21], [22], [23]
Extended Use Case Test	1	[21]
Structural Testing (without a specific criterion)	1	[4]
Functional Testing (without a specific criterion)	3	[4], [17], [29]
Systematic Functional Testing	2	[11], [30]
Extended Systematic Functional Testing	1	[30]

A. *Results of the primary question: Which comparisons have been made between test criteria?*

This question aimed at identifying primary studies that carried out comparisons between functional test criteria from any perspective. Results revealed few studies with such an objective. Among the studies analysed, only [21] and [27] make comparisons. The former compares criteria applied to object-oriented systems, whereas the latter uses both Boundary Value Analysis and Equivalence Class Partitioning (also known as Equivalence Partitioning) and compares them to other test criteria, i.e., Decision Table.

In our third inclusion criterion, which includes studies comparing structural and random testing techniques, nine studies were added to the previous two. Therefore, a total of 11 studies were selected for the primary question. Among the criteria considered of interest to our systematic mapping, Vallespir and Herbert [27] concluded that Equivalence Partitioning obtained better results than Decision Table regarding three comparative features: (i) number of defects, (ii) detection time and (iii) efficiency (quantity/time). Seo and Choi [21] concluded that Extended Use Case Test and Test Derived from Formal OCL Specifications are the most effective and suggested the combined use of them.

All studies presented in [4], [5], [7], [9], and [10] stated that, in general, Boundary Value Analysis and Equivalence Class Partitioning showed the best results regarding the number of defects detected in a short period of time. However, almost all of them agree that results depended on program type, tester experience and type of defect detected.

Similarly to studies [5] and [7], study [9] noted that up until 1997: (i) there was no consistent evidence to support that one technique for defect detection was better than another; on the contrary, current evidence suggests that every technique has its own merits; (ii) current evidence shows that functional, structural and code review testing techniques complement one another, and should be used in combination.

In summary, comparative features relevant to the research question were applied to the selected studies. However, the results obtained from the application of these features are not definitive for two main reasons: (a) tested programs are very small and simple, and (b) defects are inserted by the tester. We consider our results as contributions to knowledge pertaining to test criteria/techniques. Thus, results may be analysed as tendencies and not as conclusions, because they cannot be generalized.

B. Results of the secondary question: What is the application scenario for each functional testing criterion?

Table II shows the studies selected to answer this research question. They were classified according to study type (experiment, theoretical analysis, simulation, case study, survey) and scope. Such perspective is relevant to assess the strength of evidence, which will be discussed in Subsection IV-A.

Table III presents application scenarios for each test criterion. It lists criteria according to the number of scenarios in which they are applied. Results revealed recurring scenarios in various criteria, which shows multiplicity of scenarios and criteria (n:n – "many for many"). In other words, the studies do not identify exclusiveness between Scenario A and Criterion B. This may be regarded as positive because criteria application scope is non-restricted within the scenarios identified.

Table II. Identified test scenarios in primary studies selected

Reference	Study Type	Scope of Study
[19]	Case study	Industry
[24]	Simulation	Industry
[25]	Simulation	Laboratory
[4]	Experiment	Academy
[6]	Theoretical analysis	Laboratory
[7]	Experiment	Academy
[8]	Experiment	Industry
[9]	Experiment	Academy
[10]	Experiment	Academy
[11]	Case study	Laboratory
[12]	Case study	Laboratory
[13]	Theoretical analysis	Industry
[14]	Experiment	Industry
[15]	Simulation	Laboratory
[17]	Survey	Laboratory
[18]	Theoretical analysis	Laboratory
[20]	Case study	Industry
[21]	Experiment	Laboratory
[22]	Simulation	Industry
[23]	Case study	Industry
[26]	Theoretical analysis	Laboratory
[27]	Experiment	Academy
[28]	Simulation	Laboratory
[30]	Case study	Academy

Table III. Test criteria/technique and scenarios

Test Criterion/Technique	Test Scenario
Boundary Value Analysis	Academic/didactic system, Non safety-critical commercial information system, Aircraft operational system, Operating system utility and Embedded commercial systems
Equivalence Class Partitioning	Academic/didactic system, Non safety-critical commercial information system, Aircraft operational system and Operating system utility
Decision Table	Academic/didactic system, Non safety-critical commercial information systems and web service
Use Case Test	Video conference, Safety-critical embedded aviation system, Safety-critical commercial information system, Safety-critical financial system, Safety-critical web system and Academic/didactic system
Cause and Effect Graph	Academic/didactic system
Extended systematic functional testing	Strategic management system and Critical commercial information system
Dynamic Partitioning	Air traffic control
Extended Use Case Test	Critical financial system
Systematic Functional Testing	Operating system utility

Results regarding scenarios showed that systems were mainly tested in academic/didactic environments, to which a total of six test criteria were applied. Next, four test criteria were used in non safety-critical commercial information systems. This is due to the fact that most studies analysed (70.38%) were developed in academic environments or laboratories. However, criteria were also applied to real life settings, i.e., safety-critical scenarios, response time, robustness, as shown in studies [19], [20], [24]. Such scenarios involve embedded systems for military aircrafts, web service testing, ticket management systems (for integrated transport systems in large metropolitan areas) and electronic component testing (mobile devices, cell phones, remote controls, television).

Among test criteria, Use Case Test was most frequent in scenarios involving critical systems (five out of three scenarios). In Extended Systematic Information Systems and Random Testing, scenarios were only applied to strategic or critical systems. Cause and Effect Graph was used only in academic/didactic scenarios. The remaining criteria were mainly applied in academic/didactic scenarios or in ones involving non safety-critical systems.

Furthermore, the first five lines in Table III show that the criteria most used in the studies were applied in a variety of scenarios.

IV. Discussion

A. Strength of evidence

Assessment of the strength of evidence is a key factor for assessing the reliability of conclusions and consequent recommendations [3], [7].

There are many systems for assessing strength of evidence. For our research, we used the GRADE system (Grading of Recommendations Assessment, Development and Evaluation) for two reasons: (i) its definitions involve the main weak points of systems that classify evidence based on hierarchy, and (ii) it may be used by other software engineering researchers [3].

The GRADE system identifies four levels of strength of evidence: high, moderate, low and very low. It is determined by a combination of four elements: study characteristics, quality, consistency and directness.

In terms of study characteristics, two thirds of the studies are observational, and one third of them are experimental. Thus, the strength of evidence of the systematic mapping is low according to GRADE definitions [3].

On the topic of study quality, data analysis approaches were moderately explained in terms of study implications, credibility and limitations. In only six out of 27 studies researchers made critical analyses of their role during research. Result credibility was discussed in 85.19% of studies. A total of 88.89% of studies pondered over their limitations. Based on these results, we may conclude that studies showed moderate evidence regarding quality.

The consistency criterion was similar across studies, given that all of them applied functional testing by use of one criterion or more, individually or in a set, in a certain scenario or in comparative experiments using criteria from other testing techniques. Therefore, the strength of evidence related to consistency was high.

Next, the aim was to test objectiveness (directness). Most studies (70.38%) were carried out in academic/laboratory contexts. Regarding intervention, most studies investigated functional testing criteria and techniques, as defined during planning. Results also showed that most studies requires empirical validation through real applications. Thus, the strength of evidence ranges between moderate and low in relation to directness.

The strength of evidence of our proposed systematic map reaches a moderate level when all four aspects are combined. Therefore, future research may alter its reliability estimate.

B. Threats to Validity

According to [31], our proposed systematic map may face two threats to its validity: (i) limitations of research sources; (ii) elaboration of research questions in accordance with works in the scientific community on the same knowledge area under investigation.

Associated with the first threat is the fact that IEEExplore and ACM Digital Library indexed databases were highly used, which may have prevented the identification of relevant primary studies that were not published in any of the two sources. Related to the second threat is the fact that the scope of the primary question includes comparisons among functional criteria as well as comparisons with criteria used in non-functional techniques.

A third threat was identified: there was no evidence of objective comparisons between test criteria. Despite this, criteria were compared in relation to efficacy, cost and efficiency. However, we noted that these factors are dependent on other ones, i.e., tester experience, the type and size of the program being tested, etc.

V. Final Considerations

The present work focused on software functional testing to contribute with its assessment and evolution. A detailed study of various functional criteria was carried out through a systematic map.

The systematic map was planned based on the model elaborated by Biolchini et al. [2] and was carried out following these research questions:

- Primary research question: Which comparisons have been made between test criteria?
- Secondary research question: What is the application scenario for each functional testing criterion?

A set of 27 primary studies were investigated. Each of them provided relevant information to support conclusions which were the basis for answering our research questions.

Regarding the primary question, only two studies compared functional testing among them, which little contributed to consolidate functional criteria knowledge and practice. A total of nine studies made comparisons between functional criteria and criteria applied to other testing techniques, i.e., Structural Testing and Random Testing. These studies showed that a certain criterion is more effective in given contexts and scenarios. We may thus conclude that testing techniques and criteria complement each other and should be applied as a set to obtain more effective results during the test process. The results of such comparisons were influenced by factors such as tester experience, type and size of the program under testing and defect types in the program.

Regarding the secondary question, as a contribution to industry and practitioners in the application of testing techniques, Boundary Value Analysis was the most used test criterion because it was analysed in a larger number of scenarios. Many application scenarios of functional test criteria were identified. The academic/learning scenario was present in most of the studies analysed. The Use Case Test was the most used in safety-critical scenarios. No scenario was exclusive to any test criterion. Tester experience and creativity were essential for criteria application, even when they were not recommended in a certain scenario.

After considerations related to the research questions had been made, the primary studies were assessed according to the quality criteria defined by Ali et al. [3] to verify strength of evidence and establish the reliability level of results. We concluded that the strength of evidence of our systematic map was moderate.

Threats to validity were also identified and assessed to verify what effects they would have in our research. Furthermore, we found that there are no similar systematic reviews. However, we identified some reviews with a specific focus, i.e., Model-based testing and concurrent software testing. This study seeks to encourage further research on systematic mapping, which is able to provide more answers to our research questions and help develop their strength of evidence.

As a future work, we intend to perform a deeper analysis of data related to the second research question, trying to provide more evidences to industry and practitioners.

Acknowledgment

The authors would like to thank the Instituto de Informática (INF/UFG), and the Brazilian Funding Agencies – FAPEG, and CAPES – which support this work.

References

[1] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. New York, NY, USA: Springer Heidelberg, 2012.

[2] J. C. de Almeida Biolchini, P. G. Mian, A. C. C. Natali, T. U. Conte, and G. H. Travassos, "Scientific research ontology to support systematic review in software engineering," *Adv. Eng. Inform.*, vol. 21, no. 2, pp. 133–151, Apr. 2007, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/sWxntK

[3] M. S. Ali, M. Ali Babar, L. Chen, and K.-J. Stol, "A systematic review of comparative evidence of aspect-oriented programming," *Inf. Softw. Technol.*, vol. 52, no. 9, pp. 871–887, Sep. 2010, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/BWkt4O

[4] G. J. Myers, "A controlled experiment in program testing and code walkthroughs/inspections," *Commun. ACM*, vol. 21, no. 9, pp. 760–768, Sep. 1978, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/xuMFHS

[5] V. Basili and R. Selby, "Comparing the effectiveness of software testing strategies," *Software Engineering, IEEE Transactions on*, vol. SE-13, no. 12, pp. 1278–1296, 1987.

[6] K. Nursimulu and R. L. Probert, "Cause-effect graphing analysis and validation of requirements," in *Proceedings of the 1995 conference of the Centre for Advanced Studies on Collaborative research*, ser. CASCON '95. IBM Press, 1995, pp. 46–46, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/OqMw8U

[7] E. Kamsties and C. M. Lott, "An empirical evaluation of three defect-detection techniques," in *Proceedings of the 5th European Software Engineering Conference.* London, UK, UK: Springer-Verlag, 1995, pp. 362–383, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/VaraHr

[8] S. C. Reid, "An empirical analysis of equivalence partitioning, boundary value analysis and random testing," in *Proceedings of the 4th International Symposium on Software Metrics*, ser. METRICS '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 64–73, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/DhMfls

[9] M. Wood, M. Roper, A. Brooks, and J. Miller, "Comparing and combining software defect detection techniques: a replicated empirical study," in *Proceedings of the 6th European SOFTWARE ENGINEERING conference held jointly with the 5th ACM SIGSOFT international symposium on Foundations of software engineering*, ser. ESEC '97/FSE-5. New York, NY, USA: Springer-Verlag New York, Inc., 1997, pp. 262–277, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/rGW3aU

[10] N. Juristo and S. Vegas, "Functional testing, structural testing and code reading: What fault type do they each detect?" in *Empirical Methods and Studies in Software Engineering*, ser. Lecture Notes in Computer Science, R. Conradi and A. Wang, Eds. Springer Berlin / Heidelberg, 2003, vol. 2765, pp. 208–232.

[11] S. Linkman, A. M. R. Vincenzi, and J. C. Maldonado, "An evaluation of systematic functional testing using

mutation testing," *7th International Conference on Empirical Assessment in Software Engineering [EASE. [S.l.: s.n.]]*, 2003.

[12] C. Nebut, F. Fleurey, Y. L. Traon, and J.-M. Jézéquel, "Requirements by contracts allow automated system testing," in *Proceedings of the 14th International Symposium on Software Reliability Engineering*, ser. ISSRE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 85–96, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/r2D0BW

[13] M. Ramachandran, "Testing software components using boundary value analysis," in *Proceedings of the 29th Conference on EUROMICRO*, ser. EUROMICRO '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 94–98, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/2gi7sT

[14] K.-Y. Cai, T. Jing, and C.-G. Bai, "Partition testing with dynamic partitioning," in *Proceedings of the 29th annual international conference on Computer software and applications conference*, ser. COMPSAC-W'05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 113–116, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/z82pJ6

[15] E. L. Jones, "Automated support for test-driven specification," Phoenix, Arizona, pp. 218–223, nov. 2005.

[16] T. Murnane, R. Hall, and K. Reed, "Towards describing black-box testing methods as atomic rules," in *Proceedings of the 29th Annual International Computer Software and Applications Conference - Volume 01*, ser. COMPSAC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 437–442, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/qhltH0

[17] J. J. Gutierrez, M. J. Escalona, M. Mejías, and J. Torres, "Generation of test cases from functional requirements. a survey," in *4th Workshop on System Testing and Validation*, Potsdam, Germany, 2006, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/Cqn1B0

[18] R. M. Hierons, "Avoiding coincidental correctness in boundary value analysis," *ACM Trans. Softw. Eng. Methodol.*, vol. 15, no. 3, pp. 227–241, Jul. 2006, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/dl0JxS

[19] C. Nebut, F. Fleurey, Y. Le-Traon, and J.-M. Jezequel, "Automatic test generation: a use case driven approach," *Software Engineering, IEEE Transactions on*, vol. 32, no. 3, pp. 140–155, 2006.

[20] S. Roubtsov and P. Heck, "Use case-based acceptance testing of a large industrial system: Approach and experience report," in *Proceedings of the Testing: Academic & Industrial Conference on Practice And Research Techniques*, ser. TAIC-PART '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 211–220, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/1M1F5F

[21] K. I. Seo and E. M. Choi, "Comparison of five black-box testing methods for object-oriented software," in *Proceedings of the Fourth International Conference on Software Engineering Research, Management and Applications*, ser. SERA '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 213–220, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/1eju7r

[22] P. Zielczynski, "Traceability from use cases to test cases," On-line article, 2006, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/RqoGJ3

[23] J. Gutierrez, M. Escalona, M. Mejias, J. Torres, and A. Centeno, "A case study for generating test cases from use cases," in *Research Challenges in Information Science, 2008. RCIS 2008. Second International Conference on*, 2008, pp. 209–214.

[24] S. Noikajana and T. Suwannasart, "Web service test case generation based on decision table (short paper)," in *Quality Software, 2008. QSIC '08. The Eighth International Conference on*, 2008, pp. 321–326.

[25] K. Vij and W. Feng, "Boundary value analysis using divide-and-rule approach," in *Information Technology: New Generations, 2008. ITNG 2008. Fifth International Conference on*, 2008, pp. 70–75.

[26] P. R. Srivastava, P. Patel, and S. Chatrola, "Cause effect graph to decision table generation," *SIGSOFT Softw. Eng. Notes*, vol. 34, no. 2, pp. 1–4, Feb. 2009, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/qhYxB0

[27] D. Vallespir and J. Herbert, "Effectiveness and cost of verification techniques: Preliminary conclusions on five techniques," in *Computer Science (ENC), 2009 Mexican International Conference on*, 2009, pp. 264–271.

[28] M. Sharma and B. Chandra, "Automatic generation of test suites from decision table - theory and implementation," in *Software Engineering Advances (ICSEA), 2010 Fifth International Conference on*, 2010, pp. 459–464.

[29] M. J. Escalona, J. J. Gutierrez, M. Mejías, G. Aragón, I. Ramos, J. Torres, and F. J. Domínguez, "An overview on test generation from functional requirements," *J. Syst. Softw.*, vol. 84, no. 8, pp. 1379–1393, Aug. 2011, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/Jq63fE

[30] A. R. Vidal, "Extended systematic funcional test: A contribution in the application of black-box testing criteria," Master's thesis, Universidade Federal de Goiás, Goiânia, 2011, (in Portuguese).

[31] D. Budgen, A. J. Burn, O. P. Brereton, B. A. Kitchenham, and R. Pretorius, "Empirical evidence about the uml: a systematic literature review," *Softw. Pract. Exper.*, vol. 41, no. 4, pp. 363–392, Apr. 2011, [retrieved: Jan., 2012]. [Online]. Available: http://goo.gl/qEc82C

A Multi-Objective Technique for Test Suite Reduction

Alessandro Marchetto, Md. Mahfuzul Islam, Angelo Susi
Fondazione Bruno Kessler
Trento, Italy
{marchetto,mahfuzul,susi}@fbk.eu

Giuseppe Scanniello
Università della Basilicata
Potenza, Italy
giuseppe.scanniello@unibas.it

***Abstract*—Entire test suites are often used to conduct regression testing on subject applications even after limited and precise changes performed during maintenance operations. Often, this practice makes regression testing difficult and costly. To deal with these issues, techniques to reduce test suites have been proposed and adopted. In this paper, we present a multi-objective technique for test suite reduction. It uses information related to the code and requirements coverage, the past execution cost of each test case in the test suite, and traceability link among software artifacts. We evaluated our proposal by testing three Java applications and comparing the achieved results with those of some baseline techniques. The results indicate that our proposal outperforms the baselines and that improvements are still possible.**

***Keywords*—Regression Testing; Requirements; Testing; Test Suite Reduction; Traceability Link Recovery.**

I. Introduction

Regression testing is usually conducted after software maintenance operations to guarantee that the effect of these operations does not compromise the expected behavior of a software application. Relevant activities often conducted during regression testing [1] are: *(i)* test selection; *(ii)* test reduction (also named minimization); and *(iii)* test prioritization. These activities are technical and business relevant because they might affect the success of a software project [2]. Among the activities above, test reduction reduces the number of test cases to be executed and should preserve the capability of a test suite in discovering faults.

To reduce test suites, existing techniques are mostly based on a single dimension (e.g., code or requirements coverage). Few attempts exist to reduce test suites and apply multiple dimensions only considering *structural* information (e.g., code coverage and execution cost), thus ignoring the *functional* dimension [3][4]. Conversely, it could be relevant to reduce test suites by explicitly taking into account structural and functional information, and the time (e.g., seconds) required to execute them.

In this paper, we propose a novel reduction technique named MORE (Multi-Objective test cases REduction). It is multi-objective and selects a subset of a test suite (i.e., reduced test suite), so decreasing the testing time while preserving the capability of the suite in exercising the application and detecting faults. The technique is based on a three-dimension analysis of test cases. The *structural* dimension concerns information regarding test cases under analysis (i.e., how they exercise the application under test), while *functional* dimension regards the coverage of users' and system requirements. The last dimension is *cost* and concerns the time to execute test cases. To deal with these dimensions traceability links among software artifacts (i.e., application code, test cases, and requirements specifications) are needed. Traceability links are often not available or not up-to-date in the project documentation. Then, we exploit Latent Semantic Indexing (LSI) [5] to infer traceability links among software artifacts and to measure their strength. To assess the validity of MORE, we have conducted an experimental evaluation on three Java applications. In this evaluation, we were mainly interested in assessing whether the test suite reduced by applying our proposal may be effective and efficient as the entire test suite.

Structure of the paper. In Section II, we discuss related work, while the used traceability recovery approach is described in Section III. In Section IV, we highlight the approach for test suite reduction, while the experiment is presented in Section V. Final remarks and future work conclude.

II. Related Work

The greater part of the approaches for test suite reduction is single-objective, [1][3]. However, multi-objective techniques have been also proposed. They largely adopt evolutionary algorithms by reformulating the test suite reduction problem as an optimization problem [15][16][17]. These approaches consider either code or requirement coverage information and try balancing that information with the execution cost of test cases as follows: *(i)* explicitly optimize them as two objectives (e.g., code coverage and execution cost); *(ii)* redefine the multi-objective to a single-objective by using an optimization function that conflates more objectives into only one. For instance, Yoo *et al.* [15] showed the benefits of the Pareto-front optimality respectively for test case selection and test minimization. They, in fact, present a two-objective approach in which code coverage and execution cost are explicitly considered when conducting test selection or minimization. To reduce test suites, MA *et al.* [17] adopted an objective function that conflates code coverage and execution cost information. Furthermore, de Souza *et al.* [16] proposed the use of the Particle Swarm Optimization (PSO) algorithm that considers two objectives for test case selection: coverage of functional requirements and execution cost.

Differently from the paper discussed above, we propose a technique to reduce test suites by explicitly considering both low- (e.g., code coverage) and high-level (e.g., requirements coverage) information about the test cases, as well as their execution cost. We fill the gap between these kinds of information by using LSI [5] to automatically recover traceability links among software artifacts. Moreover, conversely to our previous work [18], we investigated the problem of reducing large test suites and, to this aim, we formulated the problem as a multi-objective optimization problem and adopted a specific implementation of the NSGA-II algorithm [9].

III. Traceability Recovery

In this paper, we applied an IR-based technique to recover traceability links: *(i)* among high-level software artifacts (i.e., application requirements and test case specifications) and low-level software artifacts (i.e., source code of the application and test case implementations); and *(ii)* between pairs of high-level software artifacts (i.e., application requirements). We use here *textual representations* of these artifacts. In the case of the test cases (implemented using special conceived frameworks, e.g., Junit), a preliminary analysis was performed to identify the application code identifiers (e.g., method and attribute names) executed by test cases. The identifiers will constitute the textual representation of the test cases. We used here LSI [5] as the IR technique. The motivation for using LSI is that it has been successfully used in the traceability recovery field [6].

A. LSI and IR-Based Traceability Recovery

LSI assumes that there is some underlying or "latent structure" in word usage that is partially obscured by variability in the word choices. To this end, a Singular Value Decomposition (SVD) is applied to a $m \times n$ matrix (also named term-by-document matrix), where m is the number of terms, and n is the number of documents in the collection. SVD can be geometrically interpreted: each term and artifact could be represented by a vector in the k space of the underlying concepts. In traceability recovery field, the similarities between two documents or between a term and a document are computed using the cosine between the vectors in the latent structure. In this work, we applied this similarity measure. The larger the value, more similar the vectors are. A value for k should be large enough to fit all the real structure in the data, but small enough so that we do not also fit the sampling error or unimportant details [5]. As default value, we used *k*=300.

Differently from typical text retrieval problems (a user writes a textual query and documents that are similar to the query are shown), in IR-based traceability recovery a set of source code artifacts (used as the query) are compared with a set of target artifacts (even overlapping). Candidate traceability links (i.e., all the possible pairs of software artifacts) are reported in a ranked list. Irrelevant links are removed using a threshold that selects only retrieved links (a subset of top links). In this work, we use the *Constant Threshold* method: 0.1 is the default value used. We used this value to limit the possibility of loosing links by considering a larger number of possible traceability links. There are also methods that do not take into account the similarity values between source and target software artifacts. For example, the method *Variable Cut Point* requires the specification of the percentage of links of the ranked list to be considered as correctly retrieved. Either relevant traceability links could be lost or irrelevant traceability links could be introduced by using methods not based on similarity values.

As in traditional IR-based traceability recovery approaches, our solution retrieves links that are either correct or incorrect so needing the human intervention to remove erroneously recovered links. To avoid that human factors may affect the experimental results, we did not perform here any analysis on the recovered links.

IV. Test Suite Reduction

We introduce our technique and the metrics used.

- **Code**. The fault detection capability of a test case and then of a test suite represents the capability to detect faults in source code. This cannot be known before executing test cases. Then, we have to resort to the "potential" fault detection capability of a test suite. It can be estimated considering the amount of code covered by test cases. A test case that covers a larger set of code statements at run-time has a higher potential fault detection capability (i.e., more faults should be revealed) than one test case that covers a smaller set of statements.

Assuming to have test case implementations (e.g., Junit test cases), we define *CCov(t)* as the amount of statements exercised during the implementation *t*:

$$CCov(t) = \sum_{s \in Statements} \begin{cases} 1 & s \in CodeCovered \\ 0 & otherwise \end{cases} \quad (1)$$

where *Statements* is the set of source code statements. *CodeCovered* is the set of statements covered by the execution of the test case *t*, *s* is a code statement of the application. Given a test suite *S* composed of ordered test cases, we defined *cumCCov(t_i)* as follows:

$$cumCCov(t_i) = \sum_{j=0}^{i-1} CCov(t_j) \quad (2)$$

where t_i is a test case of the suite. The cumulative code coverage for t_i is computed by summing the single code coverage (i.e., the code covered only by the test case) of all those test cases from t_0 to t_{i-1}.

- **Requirements**. The capability of a test case to exercise users' and/or system requirements depends on: *(i)* the amount of the requirements covered by the test case; *(ii)* the relevance of the covered requirements; and *(iii)* the existing dependency/relationship among requirements. We defined and used *RCov(t)* and a weighted variant *WRCov(t)*. *RCov(t)* is the measure of the requirements coverage for the test case *t*. This measure estimates the application requirements exercised during the execution of *t* and it is computed by counting the number of requirements exercised by the test case *t*. *WRCov(t)* measures the coverage for a test case according to predefined weights assigned to each application requirement. This coverage measure is computed as follows:

$$WRCov(t) = \sum_{r \in Reqs} \begin{cases} w_r & r \in ReqsCovered \\ 0 & otherwise \end{cases} \quad (3)$$

Reqs is the set of requirements of the application under test. *ReqsCovered* is the set of requirements covered by the execution of the test case *t*, while *r* is one of the application requirement and w_r ($0 \leq w_r \leq 1$) is the predefined weight associated to each requirement. Notice that if we consider all requirements equally (i.e., w_r=1), we resort to *RCov(t)*. The requirements weight w_r depends on the testing needs. In this work, we use as default values three weights associated to the labels *high, medium, low* [2]:

$$w_r = \begin{cases} 1 & r \in TesterRelevant_r \\ 0.5 & TesterPartialRelevant_r \\ 0 & TesterNonRelevant_r \end{cases} \quad (4)$$

where *TesterRelevant*$_r$ and *TesterPartialRelevant*$_r$ are those requirements *r* selected by the tester as relevant or partially relevant, instead the remaining requirements are *TesterNonRelevant*$_r$. However, alternative definition of the weight w_r can be considered. In fact, as in the code coverage case, the use of this weight w_r is expected to be useful to customize the measurement of the requirements coverage according to the tester's need. Hence, requirements prioritization techniques [7] could be applied to automatically identify requirements that are relevant for the tester's purposes and then to be highly weighted when measuring the coverage.

RCov(t) and *WRCov(t)* do not consider the existing relationship among requirements: all the requirements are considered equally. This issue can lead to situations in which groups of slightly connected requirements (i.e., those requirements having a limited number of related requirements) are privileged than the more connected ones. To deal with this issue, we define *WRCovD(t)*. It takes into account existing relationships among requirements. For sake of simplicity, in the following, we applied the variant only to *WRCov(t)* but the same could be done with *RCov(t)*. To compute *WRCovD(t)*, we need to measure the strength of each requirements relationship/dependency (rD). This strength is computed as follows:

$$w_{rD}(r_l, r_m) = \frac{w_{req}(r_l, r_m) + w_{code}(r_l, r_m)}{2} \quad (5)$$

where $w_{rD}(r_l, r_m)$ is the weight of the relationship in rDs between requirements: r_l and r_m; $w_{rD}(r_l, r_m)$ tends to 1 if a strong relationship exists between r_l and r_m, i.e., both textual description and implementation strongly overlap, while $w_{rD}(r_l, r_m)$ tends to 0 if no relationship exists between r_l and r_m. $w_{req}(r_l, r_m)$ and $w_{code}(r_l, r_m)$ are the weights of the relationship with respect to requirements r_l and r_m and their implementation code, and are computed as follows:

$$w_{req}(r_l, r_m) = IRSimilarity(r_l, r_m) \quad (6)$$

$$w_{code}(r_l, r_m) = \frac{overlapClasses(r_l, r_m)}{totalClasses(r_l, r_m)} \quad (7)$$

$w_{req}(r_l, r_m)$, inferred by LSI, provides an indication about the possible link between the application requirements r_l and r_m, while $w_{code}(r_l, r_m)$ computes the portion of code that is in common between the implementation of the requirements r_l and r_m.

The final requirement coverage of *t* is computed as:

$$WRCovD(t) = \sum_{r \in Reqs} w_r * (\sum_{r_l \neq r \in Reqs} w_{reqs}(r, r_l)) \quad (8)$$

where w_r is the predefined weight associate to each requirement. The weight of the dependencies between the current requirement r and the other requirements of the application are computed by the formula: $\sum_{r_l \neq r \in Reqs} w_{rD}(r, r_l)$. $WRCovD(t_i)$ is expected to give more relevance than *WRCov(t)* to the test cases covering requirements having strong relationships with a high number of other requirements, that is to the test cases exercising "key" requirements.

Given a test case t$_i \in S$, we define:

$$cumRCov(t_i) = \sum_{j=0}^{i-1} WRCovD(t_j) \quad (9)$$

The cumulative requirements coverage for the test case t_i is computed by summing the single requirements coverage (i.e., the requirements covered only by the test case) of all those test cases from t_0 to t_{i-1}.

- **Execution cost**. The execution cost of a test case can be approximated by the time required to its execution. If the implementation of the test cases is available, their execution can be profiled to collect the information about the running time. Alternatively, we can approximate the execution time by counting the number of software elements (e.g., code classes, methods) expected to be exercised by the test case. In this work, we assume to have the test implementation (e.g., Junit test cases), thus we defined *Cost(t)* as the estimated time required to execute the test case.

Therefore, given a test suite *S*, whose test cases are ordered, we computed *cumCost(t$_i$)* as the sum of the execution costs of the test cases preceding the test case *t$_i$* $\in$ *S*. The overall cost of the test cases of a suite *S* (named *Cost(S)*) is the sum of the executions of all the test cases. We then define *InverseCost(t$_i$)* as follows:

$$InverseCost(t_i, S) = Cost(S) - \sum_{j=1}^{i} Cost(t_j) \quad (10)$$

A. Measure for test reduction

For each test case t_i in the test suite S, the measures *cumCCov(t$_i$)*, *cumRCov(t$_i$)*, and *InverseCost(t$_i$)* are computed considering the position of t_i in S. Then, for each measure above, we computed the area of the curves obtained by plotting in a *Cartesian* plan the values of the metric (on X axes) with respect to the test cases in *suite*$_S$ (Y axes). To get a numerical approximation of that area, we used the *Trapezoidal* rule [8]. It computes the area of a curve as the area of a linear function that approximates that curve.

For a test suite S and each defined cumulative measure, the area (*AUC* in the following) estimates: the code coverage *AUCcumCCov(S)*, the requirements coverage *AUCcumRCov(S)*, and the execution cost *AUCInverseCost(S)*. The area indicates how fast the test suite S converges. The larger *AUC*, the better is.

B. Multi-Objective Reduction

The evaluation of all the possible test case subsets on the three dimensions could be expensive even if in case of non-large test suites. Hence, we propose the use of a multi-objective optimization to prioritize test cases according to the three identified measures. Specifically, we rely on the Non-dominated Sorting Genetic Algorithm II (NSGA-II [9]). Even if different evolutionary algorithm could be used, we resort to NSGA-II since it lets us optimize several, potentially conflicting, objectives. It has been also widely and successfully used in research work goals similar to ours [10][11].

NSGA-II uses a set of genetic operators (i.e., crossover, mutation, selection) to iteratively evolve an initial population of candidate solutions (i.e., reduced test suites). The evolution is guided by an objective function (called fitness function) that evaluates the quality of each candidate solution along the considered dimensions. In each iteration, the *Pareto* front of the best alternative solutions is generated from the evolved population. The front contains the set of non-dominated solutions, i.e., those solutions that are not inferior to any other solution in *all* considered dimensions. Population evolution is iterated until the maximum number of iterations is reached.

The Pareto front represents the optimal trade-off between the structural, functional, and cost dimensions. The tester can inspect the Pareto front to find the best compromise between having a test case ordering that balance code coverage, requirements coverage, and execution cost or alternatively having a test case ordering that maximizes one/two dimension/s penalizing the remaining one/s. This depends on the testing needs.

Specifically, the technique is set-up as follows:
1. Solution Encoding: A solution is a possible reduced test suite red_S of the application under test. This red_S represents an execution order for a subset of the test cases of the whole test suite *S*. The solution space for the test reduction problem is given by all the permutations of all the possible subsets of the test suite. A reduced test suite is represented as a sequence of integers, where each integer represents a test case identifier and the size of the reduced suite can be set-up by the tester. The maximum number of test cases per suite is a parameter of the algorithm that the tester can customize (e.g., 30% of the whole test suite).
2. Initialization: We randomly initialize the starting population by selecting subsets of test cases among all the possible of test case subsets.
3. Genetic Operators: NSGA-II resorts to three genetic operators for the evolution of the population: mutation, crossover, and selection. The standard operators typically applied for subset of (permutation-based) encoding of solutions are used. As mutation operator, we used the bit-flip mutation: one randomly chosen element of the solution is changed. The adopted crossover operator is the one-point crossover, in which a pair of solutions is recombined by cutting the two solution representations randomly chosen (intermediate) point and swapping the tails of the two cut solutions. We used binary tournament as the selection operator: two solutions are randomly chosen and the fitter of the two is the one that survives in the next population.
4. Fitness Functions: The objective is to maximize the three considered dimensions. Then, each candidate solution in the population (each reduced test suite) is evaluated by our objective function based on: $AUCcumCCov(red_S)$, $AUCcumRCov(red_S)$, and $AUCInverseCost(red_S)$. The larger these values, the faster a reduced test suite converges.

V. EXPERIMENT

To assess the validity of both the technique and the prototype, we conducted an experiment in which we compared test suites reduced with MORE against: *(i)* whole test suites (Full); *(ii)* test suites reduced according to their capability of covering the code (CC) of the target application: CC reduces a test suite S by prioritizing its test cases applying additional code coverage (additional code coverage evaluates each test case of a suite according the code portion that is uniquely covered by it [3]) and then selecting the top-ranked test cases to be part of the reduced suite [3]; and *(iii)* test suites reduced randomly (RA) [12].

A. Experimental Objects

In the study, we used three Java applications AveCalc, LaTazza and iTrust. All applications are distributed online and have been already used in the literature for different purposes [13]. AveCalc manages electronic record books for students: it has 8 classes for 1827 LOCs (excluding comments); it is distributed with 10 textual users' requirements, and 47 JUnit test cases. Latazza is a coffee maker management application: it has 18 classes for 1121 LOCs (excluding comments); it is distributed with 10 textual users' requirements, and 33 JUnit test cases. iTrust Medical Care is a medical application: it has 232 classes for 15495 LOCs (excluding comments); it is distributed with 15 textual users' requirements and with 919 JUnit test cases.

B. Procedure

For each experimental object, we applied the following experimental procedure:
1. Collecting the artifacts: requirements specifications, source code, and test cases.
2. Recovering the traceability links among such software artifacts. As mentioned before, we used the following set-up for LSI: *k*=300; *constant threshold*=0.1.
3. Applying the test reduction techniques (i.e., RA, CC and MORE) to get subsets of the whole test suite, i.e., Full. To balance the number of test cases in the reduced suites, we fixed the size of the reduced test suites (e.g., 30% of Full). Note that we ran MORE with the following set-up: *population size*=2*"test suite size"; *crossover probability*=0.9; *mutation probability*=1/"test suite size". We executed different runs of MORE considering different iterations, that is from *max iterations*=1k to *max iterations*=100k. We, moreover, executed both MORE and RA several times (4 and 20 times, respectively) and evaluated all solutions generated by them. This lets us analyze the average behavior of the techniques (reporting descriptive statistics about the obtained values). MORE has been also executed by weighting the requirements coverage (i.e., using *WRCovD* as the measure for the requirements coverage) according to a requirements prioritization defined by one tester not involved in the rest of the study.
4. Injecting faults in the source code of the application. We injected 15, 15 and 21 faults in AveCalc, LaTazza, and iTrust, respectively. This task was accomplished by an author not involved in the rest of the study. Further details are not provided for space reason (see also [18]).
5. Executing all the test suites in the faulty applications and collecting information about the different evaluation criteria.
6. Repeating the experiment considering several size of the reduced suites: 10%, 20%, 30% and 40% of Full and also after having perturbed the traceability links recovered by MORE (i.e., robustness evaluation).

C. Measures Used for the Comparison

The comparison has been performed with respect to the following evaluation criteria and metrics:

- **Size** ($Size(S)$): *What is the size of the reduced suites?* $Size(S)$ estimates the test effort required to execute the suite S. It is computed as the number of test cases of S.

- **Effectiveness** ($Effect(S)$): *What is the capability of the reduced suites in discovering (injected) faults?* $Effect(S)$ measures the capability of S to reveal (injected) faults. It is evaluated by considering two metrics: $Fault(S)$, the number of revealed faults; and $rFDC(S)$, the fault detection capability rate of S. $rFDC(S)$ is computed as follows:

$$rFDC(S) = \frac{\sum_{f \in F} \frac{FR^f(S)}{|S|}}{|F|} \tag{11}$$

$FR^f(S)$ is the set of test cases in S that reveals the fault f. F is the set of all known faults. $rFDC(S)$ gives us an idea about the capability in revealing faults of the test cases of the suite S. The higher the value of both $Fault(S)$ and $rFDC(S)$, the greater the capability to find faults of the suite S is, that indicates a highly effective suite.

- **Sensitivity** ($Sens(S)$): *What is the capability of the reduced suites of discovering faults affecting top-relevant application requirements?* $Sens(S)$ provides an indication of the capability of S in revealing faults having a high severity and relevance with respect to the application requirements, as well as the application business. $Sens(S)$ is evaluated by means of $Fault'(S)$ applied to the subset of the injected faults that affect relevant application requirements.

- **Efficiency** ($Effic(S)$): *What is the efficiency of the reduced suites in discovering faults?* $Effic(S)$ estimates the capability of S in early detecting the faults and it is measured as:

$$Effic(S) = \frac{Fault(S)}{ECost(S)} \tag{12}$$

$Effic(S)$ is the efficiency computes as the number of detected faults $Fault(S)$ divided the time spent to do it $ECost(S)$ (i.e., the time to run the test cases of the suite S). The larger the value, the more efficient the approach is.

- **Artifact coverage**: *What is the capability of the reduced suites of covering the applications artifacts?* It gives an idea about how the test suite covers both the application code ($Code\_Cov(S)$) and requirements ($Reqs\_Cov(S)$). In detail, we measure two metrics: $Code\_Cov(S)$ is measured in terms of executed code statements exercised at least once by the test cases of the suite while $Reqs\_Cov(S)$ is measured in terms of number of requirement exercised at least once by the test cases of the suite.

- **Diversity** ($Div(S1, S2)$): *How differ the reduced suites are?* $Div(S1, S2)$ estimates the difference of the test cases composing the reduced suites $S1$ and $S2$. It is measured by the Levenshtein edit distance [14] (Ld). This distance indicates the minimum number of operations (insert, delete, and replace) to transform a source string into a target string both built using the same alphabet (i.e., representing test cases of suites $S1$ and $S2$ reduced from S). The values of Ld range from 0 (the two strings are the same) to the maximum length of the two strings (the strings are completely different). Given two testing subsets (***Red1***$_S$ and ***Red2***$_S$) for a suite S with a fixed number n of test cases, Div is computed as:

$$Div(Red1_S, Red2_S) = (\frac{Ld(Red1_S, Red1_S)}{n}) * 100 \tag{13}$$

- **Robustness** ($Robu(S)$): *How "noise" in the recovered traceability links impacts on the capability of the suites reduced by MORE in revealing faults?* $Robu(S)$ measures the capability of the test reduction technique to adequately work in presence of incomplete or spurious/wrong traceability links (i.e., "noise"). It is evaluated by randomly perturbing the traceability links identified by MORE and re-computing the evaluation criteria for the obtained suites (e.g., effectiveness, efficiency).

- **Settings**: *How the MORE parameter settings can influence the obtained suites in revealing faults?* It gives an indication about how to set-up MORE to make it effective and efficient in revealing faults. With the aim of studying how MORE works in different settings we considered, in particular, different number of iterations of the evolutionary algorithm implemented by MORE and different size of the test suites reduced.

D. Results

Table I summarizes the achieved results in terms of: minimal, median, and maximal values for some of the collected measures (e.g., effectiveness, sensitivity) for the three applications. On the other hand, Figure 1 plots the number of faults revealed by each technique for the three applications. Notice that these results are for the reduced suites containing 30% of Full suites. However, similar results and plots have been collected also for reduced suites having different size, i.e., 10%, 20%, 40% of Full suites. Figure 2 shows the distribution of code coverage and discovered faults for AveCalc at increasing size of the reduced suite (i.e., from 10% to 40% of the Full suite); similar plots have been obtained for all considered metrics and applications. Finally, Figure 3 shows the distributions of discovered faults and efficiency for AveCalc by considering: (i) the reduced suite that is constituted by 30% of the Full size; and (ii) different iterations of our test reduction algorithm: 1k, 4k, 10k and 100k. Similar plots have been obtained for all metrics and applications.

- **Effectiveness**. Table I (values in bold) and the corresponding plots for AveCalc, LaTazza and iTrust in Figure 1 show that the suites reduced with MORE overcome, in most of the cases, the ones reduced by CC and RA while, in few cases, its result is comparable with the best suites obtained from CC and RA. The results achieved by CC and RA are generally worse. We observe that the capability in revealing faults of suites reduced with MORE (and using 30% of the Full test cases) is, at least, double with respect to the other reduced suites, considering the minimal number of revealing bugs per suite. In particular, the suites reduced with RA have an highly variable capability of revealing faults, with respect to those achieved by MORE. This suggests also that MORE can improve the capability of test suites reduced by CC and RA in revealing faults by explicitly optimizing them with respect to code and requirements coverage and execution time as well. However, the good results achieved in few cases by RA, in terms of revealed faults, indicates that improvements are still possible.

TABLE I. SUMMARY OF THE ACHIEVED RESULTS FOR THE REDUCED SUITES HAVING SIZE 30% OF THE SUITES: FULL

	AveCalc				LaTazza				iTrust			
	Full	RA	CC	MORE	Full	RA	CC	MORE	Full	RA	CC	MORE
Size (%)												
	100	30			100	30			100	30		
Effectiveness												
Fault_{min}	15	1	6	6	15	2	6	5	21	1	7	3
Fault_{med}	-	5	-	8	-	4	-	6	-	4	-	6
Fault_{max}	-	8	-	**10**	-	**8**	-	**8**	-	**11**	-	10
rFDC_{min}	0.053	0.009	0.05	0.05	0.044	0.01	0.05	0.04	0.0019	0.0002	0.0019	0.0007
rFDC_{med}	-	0.004	-	0.08	-	0.04	-	0.05	-	0.0014	-	0.0014
rFDC_{max}	-	0.08	-	**0.09**	-	0.07	-	**0.07**	-	**0.0026**	-	0.0024
Sensitivity												
Fault'_{min}	6	0	4	2	6	0	3	2	12	1	7	2
Fault'_{med}	-	3	-	3	-	2	-	3	-	5	-	5
Fault'_{max}	-	**4**	-	**4**	-	**4**	-	**4**	-	7	-	**10**
Efficiency												
Effic_{min}	0.83	0.8	0.95	1.5	0.62	0.8	2.2	1.9	0.075	0.011	0.076	0.041
Effic_{med}	-	1.1	-	2	-	1.5	-	2.4	-	0.059	-	0.083
Effic_{max}	-	1.7	-	**2.5**	-	2.6	-	**3.2**	-	0.132	-	**0.133**
Artifact Coverage												
Code_Cov_{min}	426	414	**426**	419	316	230	301	233	7772	4602	**7430**	4690
Code_Cov_{med}	-	420	-	424	-	284	-	296.5	-	4998.5	-	5681
Code_Cov_{max}	-	**426**	-	**426**	-	308	-	**312**	-	5422	-	6095
Reqs_Cov_{min}	7	6	**7**	6	5	3	**5**	4	**14**	**14**	**14**	**14**
Reqs_Cov_{med}	-	**7**	-	**7**	-	**5**	-	**5**	-	**14**	-	**14**
Reqs_Cov_{max}	-	**7**	-	**7**	-	**5**	-	**5**	-	**14**	-	**14**
Robustness												
Fault_{min}	15	-	-	6	15	-	-	6	21	-	-	2
Fault_{med}	-	-	-	7	-	-	-	7	-	-	-	4
Fault_{max}	-	-	-	8	-	-	-	8	-	-	-	7

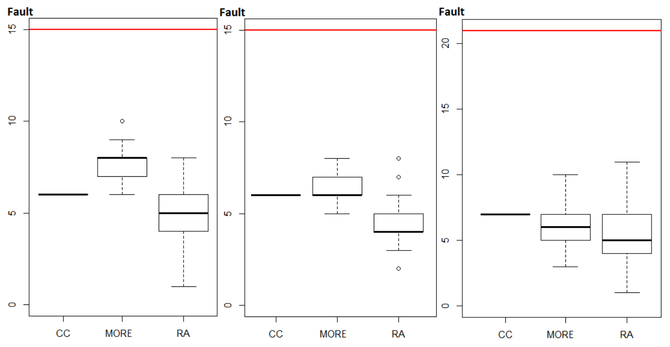


Fig. 1. Boxplots of Faults for AveCalc (left), LaTazza (center) and iTrust (right). The solid line indicates the result of Full.

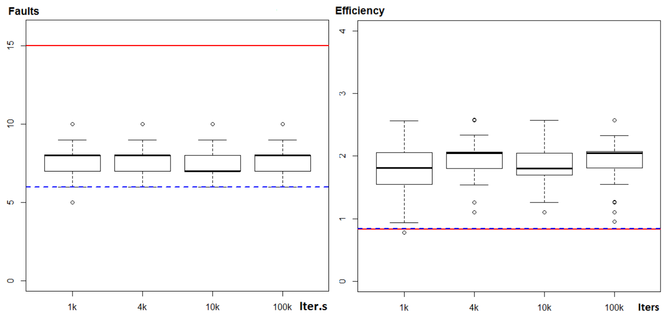


Fig. 3. AveCalc: results at increasing iterations. The solid line indicates the result of Full, the dashed one the result of CC.

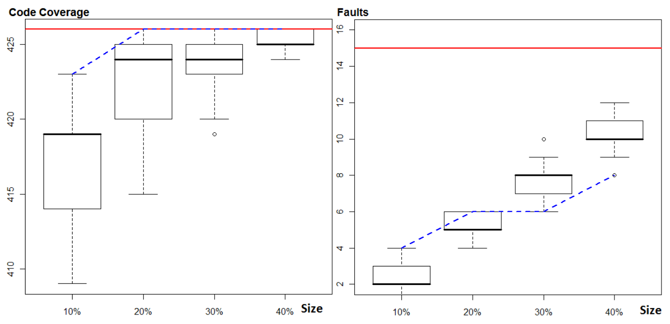


Fig. 2. AveCalc: results at increasing suite size. The solid line indicates the result of Full, the dashed one the result of CC.

- **Sensitivity**. Table I shows that the suites reduced with MORE overcome the ones reduced by CC and RA in terms of minimal number of severe faults impacting top-three relevant requirements (identified by one tester not involved in the rest of the study), and for iTrust also in terms of maximum number of revealed faults.

- **Efficiency**. Table I shows that the suites reduced with MORE always overcome all the other suites (reduced and full ones) in terms of efficiency in revealing faults, i.e., they have required less time to reveal each fault.

- **Artifact coverage**. Table I shows that the suites reduced with MORE achieved a good coverage degree of the application artifacts, i.e., code and requirements. In particular, we can observe that in the considered applications, reduced suites composed of 30% of test cases of the Full suites have the capability to cover: (i) almost all the application requirements used in the study (i.e., more than 60% of requirements); and (ii) a relevant portion of the application source code (i.e., more than 59% of requirements). By manually inspecting test suites and application requirements, we observed that the suites contain redundant test cases, that is test cases that exercise the same portion of code but using different input values and oracles. In addition, some of the used textual application requirements represent quite high-level descriptions of requirements and they do not present too many details, thus they shown high similarity with several test cases, according to LSI.

- **Diversity**. Table II shows the values collected for Div. The test suites reduced by MORE seems to be highly different from the ones reduced with the other techniques. In particular, the high value of the minimal diversity (i.e., 42%), achieved in all the applications by the suites reduced with MORE and CC, suggests a substantial difference of the composition of the test suites reduced by MORE with respect to the ones generated by the single-objective (i.e., CC) technique. While, the high value of the minimal diversity (i.e., 85%), achieved in all the applications by the suites reduced with MORE and RA, suggests that some of the suites reduced by RA are strongly

TABLE II. AVERAGE RESULTS ABOUT *DIV*

DIV	AveCalc	LaTazza	iTrust
MORE - RA	$92.4 \div 100$	$85 \div 100$	$99.1 \div 100$
MORE - CC	$42 \div 100$	$42.7 \div 100$	$98.2 \div 100$

similar to the ones generated by MORE.

- **Robustness:**. Table I shows that the suites reduced with MORE revealed less faults, on average, than the corresponding suites reduced using the actual traceability links recovered by MORE. However, for LaTazza the number of revealed faults increases of few points, this indicates the existence of traceability links incorrectly recovered. Further experimentation needs to be devoted to evaluate and detect such links.

- **Settings:**. Figure 2 shows, as example, the results of the suites reduced with MORE for AveCalc at different suite size, respectively for the code coverage measure (left figure) and for the discovered faults (right figure). Similar plots have been computed for all evaluation criteria and applications. From these plots, we observe that the suites reduced with MORE at 20,30% of Full suite size achived results almost comparable to the same Full suites, in terms of artifacts coverage, and reasonably high results in terms of effectiveness and efficiency. Conversely, the MORE suites built using less than 20% of Full performed better, in terms of revealed faults, than CC. We argue that this is mainly due to the fact that the suites reduced with MORE by considering, e.g., 10% of Full size have a quite limited coverage of the application artifacts, than CC (Figure 2-left the plot of code covered by MORE and CC). Furthermore about the technique settings, Figure 3 shows that increasing the maximum number of iterations of the evolutionary algorithm implemented by MORE does not allow achieving better results in term of discovered faults and suite efficiency (see the plots of all the three applications).

- **Final remarks**. In conclusion, the results achieved in the experiment show that: *(i)* consistently with the existing literature [15], the multi-objective optimization is overall effective in reducing test suites by balancing different dimensions and *(ii)* MORE achieves good results and it tends to outperform CC and RA, even when a non-trivial suite reduction (e.g., 20/30% of the full suite) is considered.

E. Threats to Validity

A possible threat that might affect the validity of the achieved results is represented by the injection of faults in the application code and their distribution. Different sets of faults can potentially lead to different results. To reduce this threat, one of the authors (not involved in the rest of the study) injected faults in the application code. An other issue could be also represented by the non-deterministic behavior of the reduction techniques used. To reduce these biases, we applied MORE and RA several times and then evaluated all the generated solutions to study the average trend. Finally, both the size and complexity of the considered applications may threaten the validity and the generalization of our results.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a multi-objective technique to reduce test suites. The technique reduces test suite considering the coverage of source code and application requirements, and the cost to execute test cases. An IR-based traceability recovery approach has been defined and applied to link software artifacts (i.e., requirements specifications, source code, and test cases). A reduced test suite is then determined by using a multi-objective optimization, implemented in terms of NSGA-II. Our technique has been evaluated using Java applications and results are promising. Future work is, however, needed to further assess MORE on bigger software applications and compare our solution with additional test reduction techniques.

REFERENCES

[1] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: a survey,” Software Testing, Verification and Reliability, vol. 22, no. 2, 2010, pp. 67–120.

[2] J. Karlsson and K. Ryan, “A cost-value approach for prioritizing requirements,” IEEE Software, vol. 14, 1997, pp. 67–74.

[3] D. Jeffrey and N. Gupta, “Improving fault detection capability by selectively retaining test cases during test suite reduction,” IEEE Trans. Softw. Eng., vol. 33, no. 2, Feb. 2007, pp. 108–123.

[4] S. McMaster and A. M. Memon, “Call stack coverage for test suite reduction,” in Procs. of Intern. Conf. on Software Maintenance, IEEE Computer Society, 2005, pp. 539–548.

[5] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, “Indexing by latent semantic analysis,” Journal of the American Society of Information Science, vol. 41, no. 6, 1990, pp. 391–407.

[6] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, “Recovering traceability links in software artifact management systems using information retrieval methods,” ACM Trans. Softw. Eng. Methodol., vol. 16, no. 4, 2007.

[7] I. Aaqib, K. Farhan M., and K. Shahbaz A., “A critical analysis of techniques for requirement prioritization and open research issues,” International Journal of Reviews in Computing, vol. 2, no. 1, 2009, pp. 8 – 18.

[8] K. Atkinson, An Introduction to Numerical Analysis, 2nd ed, Wiley, 1989.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Trans. Softw. Eng., vol. 6, no. 2, 2002, pp. 182–197.

[10] A. Marchetto, C. Di Francescomarino, and P. Tonella, “Optimizing the trade-off between complexity and conformance in process reduction,” in Procs. of Intern. Conf. on Search based software engineering, Berlin, Heidelberg: Springer-Verlag, 2011, pp. 158–172.

[11] Y. Zhang and M. Harman, “Search Based Optimization of Requirements Interaction Management,” in In Procs. Intern. Symposium on Search Based Software Engineering, IEEE Computer Society, 2010, pp. 47–56.

[12] A. Arcuri, M. Iqbal, and L. Briand, “Random testing: Theoretical results and practical implications,” IEEE Trans. Softw. Eng., vol. 38, no. 2, 2012, pp. 258 –277.

[13] F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, M. Ceccato, and C. A. Visaggio, “Are fit tables really talking?: a series of experiments to understand whether fit tables are useful during evolution tasks,” in Procs. of Intern. Conf. on Software Engineering, ACM, 2008, pp. 361–370.

[14] V. Levenshtein, “Binary Codes Capable of Correcting Deletions, Insertions and Reversals,” Soviet Physics Doklady, vol. 10, 1996, p. 707.

[15] S. Yoo and M. Harman, “Pareto efficient multi-objective test case selection,” in Procs. of Intern. Symposium on Software testing and analysis, ACM, 2007, pp. 140–150.

[16] L. de Souza, P. de Miranda, R. Prudencio, and F. de Barros, “A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort,” in Procs. of Intern. Con.e on Tools with Artificial Intelligence, 2011, pp. 245 –252.

[17] X. MA, Z. He, B. kui Sheng, and C. Ye, “A genetic algorithm for test-suite reduction,” in Procs. of Inter. Conf. on Systems, Man and Cybernetics, vol. 1, 2005, pp. 133–139.

[18] M. M. Islam, A. Marchetto, A. Susi, and G. Scanniello, “A multi-objective technique to prioritize test cases based on latent semantic indexing,” in Procs. of European Conference on Software Maintenance and Reengineering, IEEE Computer Society, March 2012, pp. 21 –30.

Applying Mutation Testing to ATL Specifications: An Experimental Case Study

Yasser Khan and Jameleddine Hassine
Department of Information and Computer Science
King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia
{yasera, jhassine}@kfupm.edu.sa

***Abstract*—Mutation testing is a well-established fault-based technique for assessing and improving the quality of test suites. In order to support mutation testing for model transformations, we define a set of eleven mutation operators for the Atlas Transformation Language (ATL). The effectiveness of the resulting operators, generated automatically using our prototype tool *MuATL*, is evaluated using a case study of an ATL program that refactors a given UML use case model. Our analysis shows that the proposed operators can successfully detect inadequacies in a given test suite.**

***Keywords-Model transformation*; *Model Driven Engineering*; *mutation testing*; *mutation operators*; *Atlas Transformation Language*;**

I. Introduction

Model transformations aim to automatically convert a *source* model to a *target* model based on a set of transformation *rules* [1]. A rule defines how attributes of a source object map to attributes of a target object. The source and target models must each conform to well defined *metamodels*, which specifies the language (syntax and semantics) of the models [2]. Apart from model refinement, model transformation can greatly improve several software development activities; including model refactoring, reverse engineering of models, and applying design patterns [3].

Faults in model transformations may result in defective models, and eventually defective code. Many approaches to test model transformations have been proposed in the literature. Lamari [4] used a functional testing approach based on a data partitioning technique that focuses on the structure of models in order to take into account the structural aspect of models when generating input test models. González and Cabot [5] and McQuillan and Power [6] have proposed white-box test model generation approaches for ATL model transformations. Fleurey et al. [7] investigated the problem of test data generation for model transformations and proposed the use of partition testing to define test criteria to cover the input metamodels. Fiorentini et al. [8] have proposed a uniform framework for treating metamodels, model transformation specifications and the automation of test case generation. Their proposed technique [8] is based on a black-box testing approach of model transformations to validate their adherence to given specifications. A gray-box testing technique has also been used by Bauer and Küster [9] for model transformations. Mottu et al. [10] have introduced the application of mutation testing to model transformations. The authors [10] have identified four semantic classes of faults (navigation, filtering, output model creation, and input model modification) for model transformations and they have defined a set of generic mutation operators to cover these class faults.

The widespread interest in testing model transformation programs provides the major motivation for this research. We, in particular, focus on investigating the applicability of fault-based testing to model transformations. To this end, this paper has the following purposes:

- It extends our previous work [11] on designing mutation operators for the ATL language [12], so that model transformation developers can gain the benefits of mutation testing.
- It evaluates the usefulness and the effectiveness of the proposed operators using a case study of a UML use case refactoring ATL specification.

The remainder of this paper is organized as follows. Our proposed ATL mutation testing approach is presented in Section II. Section III introduces a suite of 11 mutation operators for the ATL transformation language. In Section IV, we apply the defined mutation operators to an ATL program that refactors a given use case model. Finally, conclusions are drawn in Section V.

II. ATL Mutation Testing Approach

Mutation testing is a well-established fault based testing technique, for assessing and improving the quality of test suites. An ATL *mutation operator* defines how a particular ATL artifact is altered in order to inject a single fault. The resulting ATL program is known as a *mutant*. If a mutant is syntactically incorrect, it is considered as an *invalid* mutant.

An ATL test suite consists of a synthesis of a number test cases consisting of input models and expected models. The original ATL program (i.e., ATL Spec S in Fig. 1) and the generated mutants run on the test cases and the results are compared using an oracle. Defining a test oracle for model transformations is a challenging task [13]. ATL Mutants are generated automatically using our prototype tool *MuATL* (Mutation Toolkit for ATL). *MuATL*, a Microsoft .NET C# based tool, is inspired by MuJava (Mutation System for Java) [14] . The execution of the test suite and the oracle function are performed manually. The automation of such activities is out of the scope of this paper.

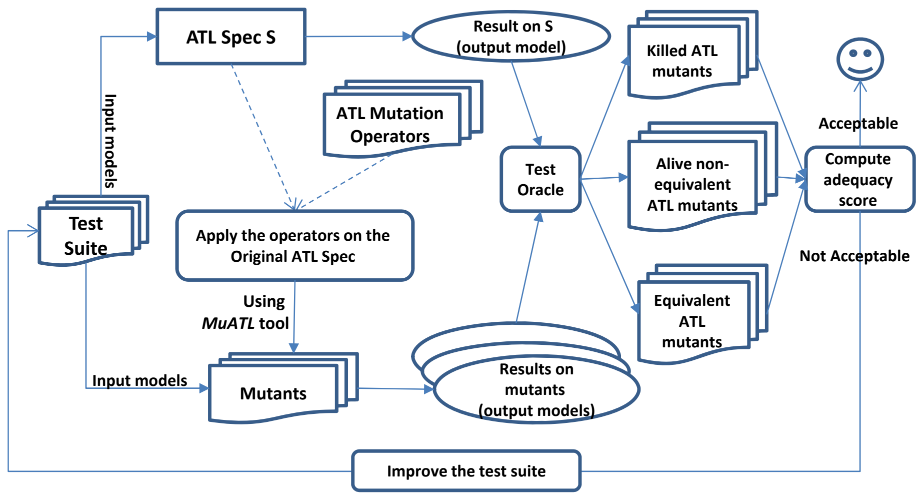


Fig. 1. ATL Mutation Process

A given test case, part of the test suite, is said to *kill* a mutant if the output model produced by the mutant is different from the expected model produced by the original ATL specification. Hence the test case is good enough to detect the change between the original and the mutant ATL program. A test case cannot distinguish between a mutant and the original ATL program if both produce the same output model(s) for the same input model. If a mutant is not killed (called *alive*) by a test suite, this usually means that the test suite is not adequate. However, it may also be that the mutant keeps the program's semantics unchanged-and thus cannot be detected by any test case. Such mutants are called *equivalent* mutants. Equivalent mutants detection is, in general, one of biggest obstacles for practical usage of mutation testing [15]. Fig. 1 illustrates our mutation testing process for the ATL language [12].

The effectiveness of a test suite is determined by running it on all mutants and computing its mutation *adequacy score*, that is the ratio of killed mutants to total number of non-equivalent mutants.

$$AdequacyScore = \frac{M_k}{M_t - M_e} \qquad (1)$$

where M_k is the number of killed ATL mutants, M_t is the total number of generated ATL mutants, and M_e is the number of ATL equivalent mutants. If the score is not acceptable, the test suite should be improved by adding additional test cases and/or modifying the existing ones.

III. ATL Mutation Operators

In this section, we briefly present the eleven proposed ATL mutation operators.

A. *Matched to Lazy (M2L)*

The M2L operator converts a matched rule to a lazy rule (which is an imperative rule). The consequence of applying the M2L operator is that a mutant rule will never be executed, since lazy rules must be explicitly invoked; thus, resulting in loss of information. If an input model contains at least one object on which the mutant rule is applicable, the corresponding M2L mutant will be killed. Otherwise, the mutant rule will not be exercised by the test case; therefore, resulting in an alive M2L mutant. An example of a mutation performed by applying the M2L operator is shown in Fig. 2(a). The M2L operator prepends the rule *AtoB* by the lazy modifier in the mutant rule *AtoB'*.

B. *Lazy to Matched (L2M)*

The L2M operator does the opposite of the M2L operator; it converts a lazy rule into a matched rule. Matched rules cannot be explicitly invoked; therefore, a runtime failure will occur when a L2M mutant rule is called. However, a L2M mutation cannot be detected if the mutant rule is not invoked during the execution. An example of a mutation performed by applying the L2M operator is shown in Fig. 2(b). The L2M operator deletes the *lazy* modifier of rule *AtoB* in the mutant rule *AtoB'*.

C. *Delete Attribute Mapping (DAM)*

The DAM operator deletes an attribute mapping from the definition of a particular rule. It is based on the CACD operator in [10]. The consequence of applying the DAM operator on a rule is that the attribute, whose mapping is deleted, will not participate in the transformation process, resulting in a loss of information. However, a DAM mutation will not be detected when the source attribute does not have a specified value. The DAM operator can be applied on matched, lazy and mapping called rules. An example of a mutation performed by applying the DAM operator is shown in Fig. 2(c). The DAM operator deletes the mapping of attribute *b2* in the mutant rule *AtoB'*.

D. *Add Attribute Mapping (AAM)*

The AAM operator adds a useless attribute mapping from a source object to a target object in a given rule. It is based on the CACA operator in [10]. The consequence of applying the AAM operator on a rule is that unnecessary complexity is

Original Program	Mutant Program
rule AtoB {	lazy rule AtoB' {
from s : A	from s : A
to t: B (	to t: B (
....	....
)	)
}	}

(a) Example of a M2L mutation

Original Program	Mutant Program
lazy rule AtoB {	rule AtoB' {
from s : A	from s : A
to t: B (	to t: B (
....	....
)	)
}	}

(b) Example of a L2M mutation

Original Program	Mutant Program
rule AtoB {	rule AtoB' {
from s : A	from s : A
to t: B (	to t: B (
b1 <- s.a1,	b1 <- s.a1
b2 <- s.a2	)
)	}
}	

(c) Example of a DAM mutation

Original Program	Mutant Program
rule AtoB {	rule AtoB' {
from s : A	from s : A
to t: B (	to t: B (
b1 <- s.a1	b1 <- s.a1,
)	b2 <- s.a2
}	)
	}

(d) Example of a AAM mutation

Original Program	Mutant Program
rule AtoB {	rule AtoB' {
from s : A (	from s : A
s.a1 > 0	to t: B (
)	b1 <- s.a1,
to t: B (	b2 <- s.a2
b1 <- s.a1,	)
b2 <- s.a2	}
)	
}	

(e) Example of a DFE mutation

Original Program	Mutant Program
rule AtoB {	rule AtoB' {
from s : A	from s : A (
to t: B (	s.a1 > 0
b1 <- s.a1,	)
b2 <- s.a2	to t: B (
)	b1 <- s.a1,
}	b2 <- s.a2
	)
	}

(f) Example of a AFE mutation

Original Program	Mutant Program
rule AtoB {	rule AtoB' {
from s : A	from s : C
to t: B (	to t: B (
....	....
)	)
}	}

(g) Example of a CST mutation

Original Program	Mutant Program
rule AtoB {	rule AtoB' {
from s : A	from s : A
to t: B (	to t: C (
....	....
)	)
}	}

(h) Example of a CTT mutation

Original Program	Mutant Program
lazy rule AtoB {	lazy rule AtoB' {
from s : A	from s : A
to t: B (	to t: B (
....	....
)	)
do {	do {
....	....
t;	}
}	}
}	

(i) Example of a DRS mutation

Original Program	Mutant Program
module A;	module A';
create OUT : UML from IN : UML;	create OUT : UML refining IN : UML;
uses B;	uses C;
uses C;	

(j) Example of a DUS mutation

Original Program	Mutant Program
module A;	module A';
create OUT : UML	create OUT : UML
from IN : UML;	refining IN : UML;

(k) Example of a CEM mutation

Fig. 2. Code examples of the proposed mutation operators

added to the output model. AAM mutants may also cause a runtime failure if the source and target attributes types are incompatible. An example of a mutation performed by applying the AAM operator is shown in Fig. 2(d). The AAM operator adds the useless mapping "*b2 <– s.a2*" in the mutant rule *AtoB'*.

E. Delete Filtering Expression (DFE)

Filtering expressions constrain the input objects on which a particular rule can be applied. If a filtering statement evaluates to true for a given input object, its corresponding rule will be executed. This can only be applied on matched rules, as they allow filtering of input objects. The DFE operator deletes the filtering statement specified in the definition of a rule. It is based on the CFCD operator in [10]. The consequence of applying the DFE operator is that the mutant rule will be executed for incorrect objects of its source type. DFE operator may cause filtering expressions of multiple rules to evaluate to true for one source instance. In this case, a runtime failure will occur. An example of a mutation performed by applying the DFE operator is shown in Fig. 2(e). The DFE operator removes the filtering expression $s.a1 > 0$ in mutant rule *AtoB'*.

F. Add Filtering Expression (AFE)

Based on the CFD operator in [10], we define the AFE operator which performs the opposite of the DFE operator. It adds an unnecessary filtering expression to a matched rule. The consequence of applying the AFE operator is that some objects of the input model will not participate in the transformation process, thus resulting in a loss of information. In order to apply the AFE operator on a rule, the source object must have at least one attribute. If this condition is satisfied, a numerous AFE mutants can be created for a given matched rule. Input Space Partitioning [16] can be applied on each source attribute to produce a set of mutant filtering expressions.

An example of a mutation performed by applying the AFE operator is shown in Fig. 2(f). The AFE operator adds the filtering expression $s.a1 > 0$ in mutant rule *AtoB'*.

G. Change Source Type (CST)

The CST operator changes the source type of a given rule. It can be applied on matched and lazy rules. The consequence of applying the CST operator is that incorrect transformations may be performed. Indeed, the application of the CST operator on a rule may cause a runtime failure if the new source type does not contain the attributes which are specified to be mapped, or if multiple rules are associated with the new source type. An example of a mutation performed by applying the CST operator is shown in Fig. 2(g). The source type of rule *AtoB* is changed from *A* to *C* in the mutant rule *AtoB'*.

H. Change Target Type (CTT)

The CTT operator changes the target type of a given rule. It can be applied on matched, lazy and mapping called rules. The consequence of applying the CTT operator is that the objects in the input model will be transformed into objects of incorrect type in the output model. An example of a mutation performed by applying the CTT operator is shown in Fig. 2(h). The target type of rule *AtoB* is changed to *C* in the mutant rule *AtoB'*.

It should be noted that CST and CTT do not produce syntacticly incorrect mutants.

I. Delete Return Statement (DRS)

The last statement of a *do* block in a mapping called rule must return the target object. It is optional to specify a return statement in the *do* block of matched and lazy rules. The DRS mutation operator deletes the return statement of a *do* block. An example of a mutation performed by applying the DRS operator is shown in Fig. 2(i). The DRS operator deletes the return statement "*t;*" of the *do* block of rule *AtoB* in mutant rule *AtoB'*.

J. Delete Use Statement (DUS)

An ATL module can import functions from a reusable library via the *uses* keyword. We define, the DUS operator which deletes an import statement from a given module. Since the ATL compiler does not check whether external functions are imported or not, the DUS operator does not produce an invalid mutant. If no external function is invoked by a test case, a DUS mutant will remain alive. An example of a mutation performed by applying the DUS operator is shown in Fig. 2(j). The DUS operator deletes the import statement of library B in mutant module A.

K. Change Execution Mode (CEM)

ATL modules can execute in two modes, *default* and *refining*. Default mode is the default execution mode of ATL transformations and it is specified by the *from* keyword. The refining mode allows developer to specify rules only for those objects that need to be transformed; remaining objects will be implicitly copied into the output model. It should be added that refining mode applies only when the source and target models conform to the same metamodel. We define the CEM operator which switches the execution mode of an ATL module from *default* to *refining* mode, or vice versa. In default mode, a CEM mutation may cause useless objects to be copied into the output model; whereas, in refining mode, it will cause loss of information. If a module contains imperative code, which is not allowed in refining mode, application of the CEM operator will result in an invalid (i.e., syntactically incorrect) mutant. An example of a mutation performed by applying the CEM operator is shown in Fig. 2(k). The CEM operator changes the execution mode of module *A* to refining mode in the mutant module *A'*.

IV. Case Study: UML Use Case Model Refactoring

The case study pertains to an ATL module, which implements a use case model refactoring. This refactoring is based on use case antipattern *a1*, which is introduced in [17]. Antipattern *a1* occurs when an actor is associated with a generalized use case in order to enable indirect access to a framework of services, which are implemented by specialized use cases. A generalized use case is often incomplete because it contains parts of common behavior required by the specialized use cases. Therefore, initiation of such a generalized use case will result in incomplete meaningless behavior. A given use case is involved in this antipattern if it:

- is a *concrete generalized* use case
- neither *includes* nor *extends* any use case
- not *extended* by any other use case
- is directly or indirectly associated with an actor

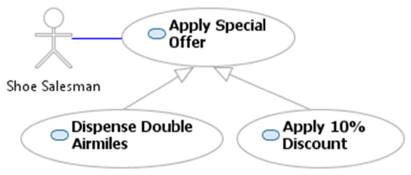
For a given input use case model, the transformation detects the model elements involved in *a1*, and performs the *ConcreteToAbstract* refactoring, which converts the *generalized* use case to an *abstract* use case. The semantics of *abstract* use cases are similar to the semantics of an abstract entity in the OO paradigm. Setting a use case as *abstract* indicates that it cannot be solely performed. Therefore, one of the *specialized* use cases will be performed. This guarantees that a complete and meaningful service will be delivered to the actor. If *a1* is not detected, the refactoring is not performed. Fig. 3 shows the subject ATL module, which is implemented in refining mode. It references three reusable libraries: *UseCase*, *Association*, and *Actor*. The filtering expression specified in the *from* clause of matched rule *AbstractGeneralizedUC* implements the detection conditions for *a1*. If a use case satisfies all of these detection conditions, its *isAbstract* property is set.

The case study contains 9 test cases which satisfy the Correlated Active Class Coverage (CACC) criteria [18], a logic coverage testing criteria that tests individual clauses in a logical expression. Each test case includes the input model and the expected output model. For instance, Fig. 4 and Fig. 5 illustrate the input model and the expected output model relative to test cases TC1 and TC2, respectively. In the input model of TC1, use case *Apply Special Offer* is involved in antipattern *a1*; therefore, it is set abstract in the output model. It should be noted that the antipattern *a1* is not detected in TC2; hence, no refactoring is performed.

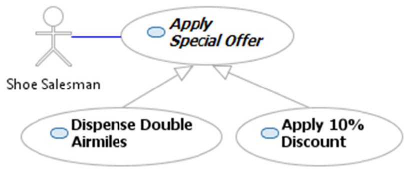
The proposed mutation operators are automatically applied on the subject module using our prototype tool MuATL, and

```
module ConcreteToAbstract;
create OUT : UML refining IN : UML;

uses UseCase;
uses Association;
uses Actor;

rule AbstractGeneralizedUC {
  from s: UML!UseCase (
    s.isGeneralization() and
       s.isConcrete() and
       not (
      s.isIncluder() or
      s.isExtension() or
      s.isExtended()
        )
       and (
         (s.isAssociatedWithActor() and
          not s.isIncluded()) or
      s.isIndirectlyAssociatedWithActor()
        )
  )
  to t: UML!UseCase (
    isAbstract<-true
  )
}
```

Fig. 3. Excerpt of the Use Case refactoring model transformation



(a) Input Use Case Model



(b) Expected Output Use Case Model

Fig. 4. Input and expected output models of TC1

result in 47 mutant modules. In addition to the proposed operators, the Conditional Operator Replacement (COR) [16], Unary Operator Deletion (UOD) [16], and the Non-Void Method Call (NVMC) [19] operators are also applied. These additional operators are used because they will target the filtering expression of rule *AbstractGeneralizedUC*.

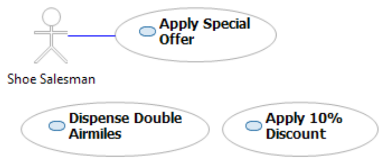
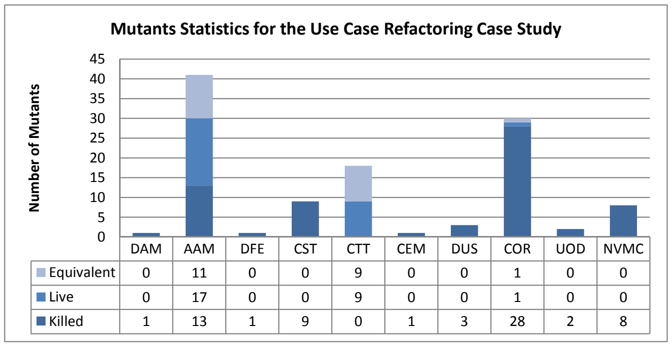


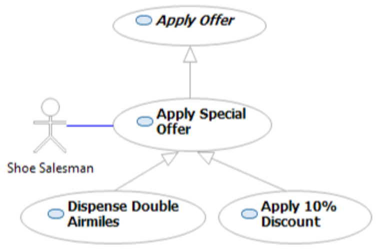
Fig. 5. Input and expected output models of TC2 (they are the same)

The rule *AbstractGeneralizedUC* contains 6 unmapped source attributes (*name*, *isAbstract*, *include*, *extend*, *generalization*, *subject*) and 5 unmapped target attributes (*name*,

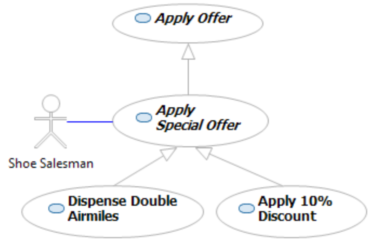


	DAM	AAM	DFE	CST	CTT	CEM	DUS	COR	UOD	NVMC
Equivalent	0	11	0	0	9	0	0	1	0	0
Live	0	17	0	0	9	0	0	1	0	0
Killed	1	13	1	9	0	1	3	28	2	8

Fig. 6. Live, killed, and equivalent mutants for the *ConcreteToAbstract* model transformation program

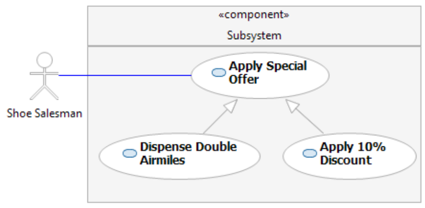


(a) Input Model for TC10

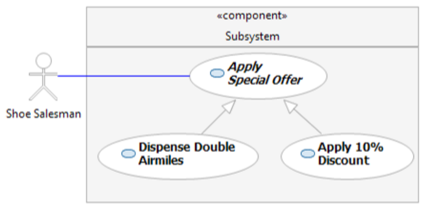


(b) Expected output Model for TC10

Fig. 7. Input and expected output models for TC10



(a) Input model for TC11



(b) Expected output model for TC11

Fig. 8. Input and expected output models for TC11

include, *extend*, *generalization*, *subject*); therefore, the application of the AAM operator resulted in 30 mutants. One DAM mutant was created for the mapping statement "*isAbstract <- true*". 10 source classes, and 10 target classes participate in the model transformation; therefore, 9 CST and 9 CTT mutants are created. It should be noted that these sets of source and target classes are the same. One DFE mutant corresponds to the filtering expression of *AbstractGeneralizedUC*. The AFE operator could not be applied on *AbstractGeneralizedUC* because it already contained a filtering expression. The M2L and L2M operators are also not applicable because the subject module is specified in refining mode. The module imports 3 reusable libraries; therefore, a DUS mutant is created for each import statement.

The results of the mutation analysis, presented in Fig. 6, reveal that 66 mutants are killed by the 9 test cases, and 27 mutants are kept alive. 1 DAM, 13 AAM, 5 CST, and 3 DUS mutants are killed as a result of runtime failures. 1 DFE, 4 CST, 1 CEM, 28 COR, 2 UOD, and 8 NVMC are killed because they produce incorrect output models. The 9 live CTT mutants are equivalent mutants; they cannot be killed by any test case. The single live COR mutant resulted in errors states for several test cases; however, these error states did not propagate into a failure. Moreover, for this mutant, no test case can be designed which will result in a failure; therefore, it was concluded as equivalent.

The nine test cases give an adequacy score of 91.67%. The obtained results show that the AAM operator determined inadequacies in the subject test suite. The 6 live non-equivalent AAM mutants (i.e., 17-11 = 6) can be killed by adding new test cases. We add TC10 and TC11, each of which kills 3 live AAM mutants, to the subject test suite. This enhanced test suite gives a 100% adequacy score. The input models of TC10 and TC11 are shown in Fig. 7(a) and Fig. 8(a), respectively.

V. Conclusions

In order to support mutation testing for ATL language, we have defined a set of eleven mutation operators. Our approach has been validated using a use case model refactoring program. The results have shown that the operators successfully detected inadequacies in the subject test suite.

As a future work, we are planning to further enhance our prototype tool *MuATL* to include a test case execution engine and a test oracle. In addition, we aim at conducting an empirical study to better assess the usefulness and the effectiveness of the proposed ATL operators.

Furthermore, we will investigate the addition of mutation operators of traditional programming languages that are relevant to ATL. The idea of mutation testing will also be explored for other model transformation languages, such as QVT, Tefkat, and Epsilon.

Acknowledgment

The authors would like to acknowledge the support provided by the Deanship of Scientific Research at King Fahd University of Petroleum & Minerals (KFUPM) for funding this work through project No. IN121009.

References

[1] A. G. Kleppe, J. Warmer, and W. Bast, *MDA Explained: The Model Driven Architecture: Practice and Promise*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003.

[2] J. M. Favre, "Towards a basic theory to model model driven engineering," in *In Workshop on Software Model Engineering, WISME 2004, joint event with UML2004*, 2004.

[3] S. Sendall and W. Kozaczynski, "Model transformation: The heart and soul of model-driven software development," *IEEE Softw.*, vol. 20, no. 5, pp. 42–45, Sep. 2003.

[4] M. Lamari, "Towards an automated test generation for the verification of model transformations," in *Proceedings of the 2007 ACM symposium on Applied computing*, ser. SAC '07. New York, NY, USA: ACM, 2007, pp. 998–1005.

[5] C. A. González and J. Cabot, "Atltest: a white-box test generation approach for ATL transformations," in *Proceedings of the 15th international conference on Model Driven Engineering Languages and Systems*, ser. MODELS'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 449–464.

[6] J. A. Mc Quillan and J. F. Power, "White-Box Coverage Criteria for Model Transformations." in *1st International Workshop on Model Transformation with ATL*, Jul 2009, p. 63.

[7] F. Fleurey, J. Steel, and B. Baudry, "Validation in model-driven engineering: testing model transformations," in *Model, Design and Validation (MoDeVa 2004), Rennes, France*, nov. 2004, pp. 29 – 40.

[8] C. Fiorentini, A. Momigliano, M. Ornaghi, and I. Poernomo, "A constructive approach to testing model transformations," in *Proceedings of the Third international conference on Theory and practice of model transformations*, ser. ICMT'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 77–92.

[9] E. Bauer and J. M. Küster, "Combining specification-based and code-based coverage for model transformation chains," in *Proceedings of the 4th international conference on Theory and practice of model transformations*, ser. ICMT'11. Springer-Verlag, 2011, pp. 78–92.

[10] J.-M. Mottu, B. Baudry, and Y. Le Traon, "Mutation analysis testing for model transformations," in *Proceedings of the Second European conference on Model Driven Architecture: foundations and Applications*, ser. ECMDA-FA'06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 376–390.

[11] Y. Khan and J. Hassine, "Mutation operators for the atlas transformation language," in *Proceedings of the 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, ser. ICSTW '13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 43–52. [Online]. Available: http://dx.doi.org/10.1109/ICSTW.2013.13

[12] F. Jouault, F. Allilaire, J. Bézivin, and I. Kurtev, "ATL: A model transformation tool," *Sci. Comput. Program.*, vol. 72, no. 1-2, pp. 31–39, Jun. 2008.

[13] J.-M. Mottu, B. Baudry, and Y. Le Traon, "Model transformation testing: oracle issue," in *IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW)*, april 2008, pp. 105 – 112.

[14] Y.-S. Ma, J. Offutt, and Y. R. Kwon, "Mujava: an automated class mutation system: Research articles," *Softw. Test. Verif. Reliab.*, vol. 15, no. 2, pp. 97–133, Jun. 2005.

[15] Y. Jia and M. Harman, "An analysis and survey of the development of mutation testing," *IEEE Trans. Software Eng.*, vol. 37, no. 5, pp. 649–678, 2011.

[16] P. Ammann and J. Offutt, *Introduction to Software Testing*, 1st ed. New York, NY, USA: Cambridge University Press, 2008.

[17] M. El-Attar and J. Miller, "Improving the quality of use case models using antipatterns," *Software & Systems Modeling*, vol. 9, no. 2, pp. 141–160, 2010.

[18] P. Ammann, J. Offutt, and H. Huang, "Coverage criteria for logical expressions," in *14th International Symposium on Software Reliability Engineering), 17-20 November 2003, Denver, CO, USA*, ser. ISSRE '03. Washington, DC, USA: IEEE Computer Society, 2003, pp. 99–107.

[19] PIT, "PIT mutation testing," http://pitest.org/quickstart/mutators/, last accessed, August 2013.

The Use of Experimentation Packages for Evaluating the Quality of Mobile Software Products

Auri Marcelo Rizzo Vincenzi
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: auri@inf.ufg.br

Gilcimar Divino de Deus
Departamento de Computação
Pontifícia Univ. Católica de Goiás, PUC-GO
Goiânia-GO, Brazil
e-mail: gyngil@gmail.com

João Carlos da Silva
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: jcs@inf.ufg.br

Plínio de Sá Leitão-Júnior
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: plinio@inf.ufg.br

José Carlos Maldonado
Inst. de Ciências Matemáticas e de Computação
Universidade de São Paulo, USP
São Carlos-SP, Brazil
e-mail: jcmaldon@icmc.usp.br

Márcio Eduardo Delamaro
Inst. de Ciências Matemáticas e de Computação
Universidade de São Paulo, USP
São Carlos-SP, Brazil
e-mail: delamaro@icmc.usp.br

***Abstract*—Mobile devices are becoming more and more common. Embedded in these devices are different mobile applications, making the devices more useful and popular. The quality of such applications is increasingly becoming a problem. Several techniques have emerged to assess software quality. In this paper, an experimentation package is proposed to evaluate some of the well-known software testing criteria on detecting faults in mobile software. This paper presents the results obtained after three replications of the proposed package. Based on statistical analysis, it was possible to arrive at statistical equivalences and differences between the evaluated criteria. This can help people concerned to establish testing strategies for mobile software.**

***Keywords*-experimental package; software testing; ubiquitous application**

I. Introduction

When a customer orders an information system, he/she lists some characteristics or requirements and he/she searches for quality in each item of the list. Software engineers must aim at quality during all the development process. According to IEEE, software product quality is defined as: "The degree to which a system, component or process meets the specified requirements and the needs or expectations of the client" [1].

Standards such as ISO 9000, 9001, and 9002 deal with quality management. One of the requirements of these models is Verification and Validation (V&V). In other words, it is necessary to determine if the product is being produced correctly, if this product meets its requirements and if it responds as expected. Software testing is largely responsible for ensuring the quality of a software product and it is one of the most common activities in software validation.

Several techniques have been adopted to expose faults in software products. *Ad-hoc* testing is based on the experience of the tester that executes a set of test cases he/she believes enough to ensure quality. A more systematic way of carrying out testing is to employ the best known functional and structural techniques.

With the functional technique, a program is tested from the user's point of view. The component being tested is considered as a black box, whose implementation details are not known, inputs are supplied, and results are compared against the expected ones. On the other hand, the structural technique, also known as a white box test, determines test cases based on implementation aspects and helps detect logical and programming faults.

In 2012, there were around 256 million cell phones in Brazil [2]. Their processing power, transmission speed, and other technological characteristics allow information handling by systems in mobile devices. It is very important for projects to be developed, which focus on improving testing strategies and applying them to the mobile environment.

One problem with mobile devices is the difficulty in testing applications in the device itself (real environment). Development and testing phases, in general, take place using emulators on desktop computers. It is extremely important for applications to be tested in their real environment, since errors may occur and be camouflaged by emulators due to their memory and processing limitations. Java Bytecode Understanting Testing/Micro-Edition (JaBUTi/ME) is a tool developed in this context, which supports the testing of Java ME software in both emulators and real devices [3].

This paper presents the results collected after three replications of an experimentation package created with the purpose of analyzing and comparing three testing techniques for mobile devices: *ad-hoc*, functional (focusing on boundary analysis and equivalence partitioning), and structural (mainly All-Nodes and All-Edges criteria [4]).

An experimentation package is a controlled and systematic way of carrying out experiments in several stages, making it possible to incrementally obtain a quantity of statistically significant data. In addition, the availability of an experimentation package allows the same study to be carried out by different people, in different places, with different cultures. This makes it possible to update these data over time, increasing the statistical database and increasing the confidence on the quality of obtained results.

This paper shows the database status after the third replication of the experimental package (investigating *ad-hoc*, functional, and structural testing techniques), and the adaptation of the JaBUTi/ME tool to support one of the testing techniques. Related works are described in Section II. In Section III, the main characteristics of the JaBUTi/ME tool are described, along with the testing criteria it supports. The experimentation package used in the replications is detailed in Section IV. Section V presents the experiment description, including the statistical data analysis. Section VI presents the conclusion of this study and future research directions.

II. Related Studies

Some studies in the literature have discussed mobile applications testing and the majority of them applies black-box testing technique without comparisons with structural testing.

Malevris [5] presented a method to effectively perform structural testing in Java programs. The proposed methods intend to generate a set of feasible paths and automatically generate test data to traverse such paths. Symbolic execution is used to identify feasible paths and the results show that, in general, the proposed methods avoid the generation of infeasible paths and ensure high coverage of the generated paths. No comparison with additional testing criteria is provided.

Pocatilu [6] focuses on the aspects related to unit testing in mobile applications based on Java ME. Emulators are used to run test cases written according to the JUnit testing framework. The author concludes that unit testing does not have to be limited to the JUnit framework, and other methods and techniques shall be used, such as the ones proposed in our evaluation.

Hu and Neamtiu [7] propose an approach for automating the testing process for Android applications. The first step was to understand the nature and frequency of bugs affecting graphical user interface (GUI) of Android applications. Following, they proposed an automated test generator for detecting these GUI bugs. The approach is based on feeding the application with random events, instrumenting the Android Virtual Machine, registering log/trace files, and analyzing them post-run. In that work, no structural testing criteria was employed to evaluate the quality of the generated test data.

In our work, we evaluate three different testing criteria considering the coverage and fault detection capability of their generated test set.

III. JaBUTi/ME and Mobile Devices

Testing without a tool increases the chance of human mistakes, and lowers productivity in test execution and analysis of results. Many tools have been produced, and each is focused on the use of one or more criteria. Java Bytecode Understanding and Testing (JaBUTi) [8] is one such tools. It explores structural testing criteria, which help creating test cases that exercise specific parts of the code.

Among the various resources offered by JaBUTi, one of the most important is the support in the coverage of bytecode-based Java programs. In others words, JaBUTi performs all computations for the Java structural test directly on bytecode, not on program source.

Java Bytecode Understanting Testing/Micro-Edition (JaBUTi/ME) is a version of JaBUTi that supports the structural testing of Java ME programs [3]. It explores the same resources as the original version and complements the original version with resources that allow program test in real mobile devices or emulators. Among the customizable resources in this version are the different code instrumentation mechanisms offered, which make it possible for the real application to communicate with the test server in accordance with memory and connectivity restrictions imposed by the different types of mobile devices, as shown in Figure 1.

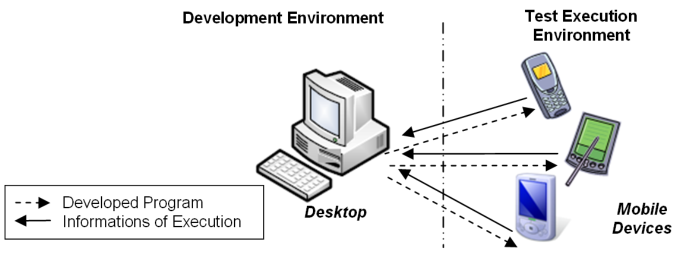


Figure 1. Environment cross platform

Program instrumentation is an essential activity for applying structural testing, making it possible to capture information about code coverage during test case execution. When a code is being instrumented, a call to a method responsible for identifying and storing information about which section of code has been executed is added to the bytecode. This information is later sent to the test server which computes the resultant coverage with respect to each testing criterion.

Since the development of JaBUTi/ME, a series of experimental studies was carried out aiming at evaluating whether its characteristics help the test of Java ME products. In this context, the focus of this study is to execute tests of programs developed for mobile devices using JaBUTi/ME. The creation of an experimentation package allows the experiments to be executed in a controlled environment and to be replicated by other researchers also interested in this research area.

The replications of one particular package help to improve the collected data, to increase the sample size and to allow more reliable conclusions. One reason for carrying out the replications was the physical impossibility of executing the entire experimentation package with a large sample of subjects. This was due to the limited number of places in software laboratories available. The replications in this study serve to increase the level of confidence in the collected data. They also help to show which technique, *ad-hoc*, functional or structural is more efficient in detecting faults in Java ME programs for the creation of new test cases and program code coverage. Another intention was to evaluate whether the resources offered by the tool are useful for testing Java ME programs in real devices and emulators. Due to the unavailability of a sufficient number of real devices, that replication was carried out using mobile device emulators.

IV. Experimentation Packages

This section describes how the experimental study using the JaBUTi/ME tool in replications was conducted. The purpose was to evaluate the three techniques mentioned earlier and their suitability for testing mobile device applications. This also made it possible to evaluate the benefits the criteria supported by the tool offer to the tester.

The goal of this study is to contribute to the development of an incremental test strategy with the support of a testing tool that can be used to improve the quality of software products and information systems used in mobile devices. Considering the increasing demand for mobile device software, the results of this study may significantly contribute to evaluating of testing techniques and to increasing in the quality of mobile software products.

A. Experimentation Package for JaBUTi/ME

The experimental study follows the process described by Wohlin et al. [9]. This experimentation package is defined and organized in the following way:

- **Definition**: Structural Test of Java ME Software in Mobile Devices Using JaBUTi/ME.
- **Context**: This experiment is an example of software engineering and, more specifically, of software testing. A specific tool, JaBUTi/ME, which was created for the structural testing of Java ME programs.
- **Hypotheses**: The following hypotheses may or may not have been valid after the experiment has been carried out.
  - Null Hypotheses:
    * $H_{0,1}$ - The structural technique, supported by JaBUTi/ME tool, detected the same number of faults as the *ad-hoc* or functional techniques;
    * $H_{0,2}$ - The structural technique, supported by JaBUTi/ME tool, obtained the same percentage of coverage as the *ad-hoc* or functional techniques;
    * $H_{0,3}$ - The structural technique, supported by JaBUTi/ME tool, did not contribute to the creation of new test cases.
  - Alternative Hypotheses:
    * $H_{1,1}$ - The structural test, supported by JaBUTi/ME tool, detected different number of faults obtained when compared to the *ad-hoc* or functional technique;
    * $H_{1,2}$ - The structural technique, supported by JaBUTi/ME tool, obtained a different percentage of coverage when compared to the *ad-hoc* or functional techniques;
    * $H_{1,3}$ - The structural technique, supported by JaBUTi/ME tool, contributed to the creation of new test cases which had not previously been identified by either the *ad-hoc* or functional test.
- **Dependent Variables**:
  - Program complexity;
  - Number of defects revealed;
  - Coverage percentage;
  - Number of new test cases.
- **Independent Variables**:
  - *Ad-hoc* technique;
  - Functional technique;
  - Structural technique;
  - Selected programs.
- **Participants**: Sixty people with computer science and Java programming knowledge participated in the experiment as subjects. The only prerequisite to participate in the experiment is a basic knowledge of Java programming. Participants should be able to recognize commands, programming structures, loops, and so on. No software testing knowledge was required.
- **Experimental Project**: Four Java ME programs were selected for the experiment. The factorial-fractional randomized technique [10] was used to assign to each subject a particular testing technique and a program to be tested. One of these programs was used for teaching functional and structural techniques. Participants used the other three to run the experiment. The participants identification by their names was not relevant for the object of the experiment. Participants were grouped merely as a way of dividing the same program among a given number of students. The information was collected and evaluated individually. It is important to mention that the programs were divided equally among the groups.

The experiment was carried out over three non consecutive days. An hour of training was provided for each technique. Later, the participants had an hour and a half to apply "hands on" the technique in one of the selected programs. The laboratory had 20 desktop computers with the Linux operating system, Java 6.0, Eclipse, Wireless Tool Kit 2.5, EclipseME, and the JaBUTi/ME tool.

The programs were selected from software repositories such as http://www.sourceforge.net and http://code.google.com. Twenty programs were pre-selected based on the availability of source code and program complexity, of which the four most complex were chosen.

All these programs were previously instrumented using JaBUTi/ME to make it possible to collect trace data during the program execution, even when the *ad-hoc* or functional technique is used to generate test cases. The execution was monitored and code coverage could be evaluated later in relation to the structural criteria implemented by JaBUTi/ME. It is important to point out that the same tool was used to evaluate the three techniques. Figure 2 shows the process of executing instrumented software and how coverage information was collected. Additionally, each subject should also fill out a form indicating when a given test case detects a fault.

- **Instrumentation**: In this stage, the forms, software, and laboratory environment for carrying out the experiment was prepared.

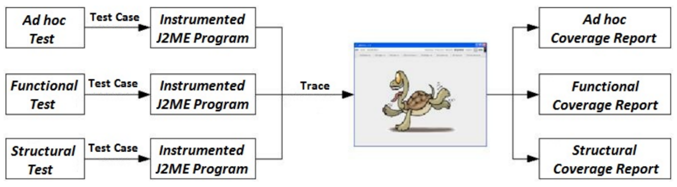

Ad hoc Test
Test Case
Instrumented J2ME Program
Functional Test
Test Case
Instrumented J2ME Program
Structural Test
Test Case
Instrumented J2ME Program
Trace
Ad hoc Coverage Report
Functional Coverage Report
Structural Coverage Report

Figure 2. Monitoring scheme outline

Four forms were prepared to be filled out by the subjects: Form 1 – Group Formation; Form 2 – Test Cases; Form 3 – Suggestions; and Form 4 – Course Evaluation. These forms and all the data collected may be obtained by contacting the corresponding author.

The four most complex pre-selected programs were chosen for the experiment. Table I presents the name of the programs used for data collection, the average maximum cyclomatic complexity of their methods, and a brief description of each.

TABLE I. SELECTED PROGRAMS AND COMPLEXITY

Id.	Name	Complexity	Description
P1	AntiPanela	3.87	Registers soccer players and performs team drawings based on the number of players, avoiding favoritism.
P2	CarManager	5.52	Monitors and manages motor vehicle fuel expenses.
P3	CódiceFiscale	6.17	Checks the validity of or generates the Italian “Codice Fiscale” tax ID.

Programs P1, P2, and P3 were used by participants to apply *ad-hoc*, functional, and structural techniques. Their order and distribution are defined in Table II. A fourth program, called BMI, which calculates Body Mass Index based on height and weight and classifies an individual according to obesity level, was used for training participants in functional and structural techniques and tools.

To assess the quality of the resulting test set on detecting faults, ten faults were artificially seeded to each program based on the concept of mutation [11]. Faults were related to variable initialization, computations, control flow, interface, and data structure. After inserting the faults, programs were compiled and instrumented using JaBUTi/ME resources to make it possible to monitor test case execution, and later to analyze their coverage in relation to the criteria supported by the tool.

- **Evaluation**: For all programs, the evaluation was based on Form 2 – Test Cases, which contains information about test case execution (faults found).
- **Preparation**: Materials and instructions for participation in the experiment were distributed. It is important to demonstrate what is really taking place as the experiment was conducted. The BMI software was chosen for teaching all of the techniques and for running programs in mobile device emulators. This software was not used for collecting information from the participants.
- **Execution**: This is the task of executing what was planned in the estimated time and documenting any deviation that could change or affect the objective of the experiment. Program specifications were also explained to the participants, so they could become familiarized with the programs under testing.
- **Data Validation**: At the end of the application of each technique by the participant, the entire project (including the trace file) must be labelled and sent to the organizing commission, ensuring that the generated data of each participant was correct.
- **Analysis and Interpretation**: Immediately after experiment and replication data had been collected, the information was cross-checked and analyzed in order to evaluate the hypotheses defined in the experimentation package.
- **Presentation and Packaging**: This paper intends to group the data of these three replications.

V. EXPERIMENT DESCRIPTION

The proposed experimentation package was replicated three times. The information collected after each replication strengthen and increase the entire experiment’s sample size.

An introduction about software testing showing the importance of testing, the role of the tester, the main kinds of tests, unit, integration, and system tests were explained to and discussed with the participants during training. The students were then randomly assigned to 6 groups. Once the groups were defined, a Java ME program together with its respective specification text were distributed to each group. The objective on the first day was to find the largest number of faults in the programs in accordance with each individual’s knowledge of software construction and testing, i.e., using the *ad-hoc* technique. The distribution of programs to the groups is shown in Table II.

TABLE II. GROUP, PROGRAM AND TECHNIQUE DISTRIBUTION

Technique/Group	G1	G2	G3	G4	G5	G6
ad-hoc	P1	P3	P2	P1	P3	P2
Functional	P2	P1	P3	P2	P1	P3
Structural	P3	P2	P1	P3	P2	P1

G – Group; P – Program;

After executing the *ad-hoc* technique, on the second day the participants received training concerning functional test technique criteria. New programs and their specifications were distributed to each group. The groups were again asked to apply the knowledge they had acquired on functional testing to carry out tests on the second program.

After the tests using the functional technique were run, on the third day participants were trained in the structural technique and use of the JaBUTi/ME tool. After this, the third and final distribution of programs was carried out and students applied structural technique concepts in running structural tests.

During the execution of the tests using any of the techniques, the participants recorded any nonconformity they found. To conclude the experiment, they were asked to fill out a form with suggestions for improvement and their individual evaluation of the course. It is important to emphasize that all students were required to test all three programs using the three different testing techniques.

A. Data Analysis

The experimentation package was prepared to capture coverage information for the programs under testing, regardless of applied technique. The JaBUTi/ME tool was used to read the data from the executed tests of each subject. The tool supports four testing criteria: All-Nodes, All-Edges, All-Uses, and All-Potential-Uses [8]. The training focused on the first two criteria, known as control flow criteria (All-Nodes and All-Edges). However, all the criteria cited above, including data flow, were analyzed by measuring the coverage of the tests in relation to these test criteria. It is important to point out that non-executable test requirements produced by the above mentioned criteria were not identified. In addition, test execution time was limited to one hour and a half. Therefore, it may be that the maximum coverage of 100% was not achieved due to these requirements and time constraints. However, since the objective was to compare which test set covered more testing requirements, the maximum obtained coverage of any test set is sufficient to establish this relationship. Tables III and IV synthesize these data.

The cumulative data after the third replication shows that the generated test set from the structural technique achieved the highest coverage of all the programs tested, and, for this set of programs, the standard deviations of the three techniques were very close (see Table IV). These data show that the values presented do not cluster around the mean and that the structural technique demonstrates better coverage for software testing in a mobile device context. Figures 3 to 5 show the coverage evolution of each testing criterion supported by JaBUTi/ME for each program under analysis. Observe that structural testing test set achieved the highest coverage in all three programs.

The structural and *ad-hoc* techniques detected more faults (see Table IV and Figure 7). Although the numbers are small in comparison with the number of faults inserted, it is important to point out that program coverage was not complete and that test execution time was a criterion in creating the experimentation package. This suggests that there is a tendency for increasing the number of detected faults as coverage also increases, which would only be possible if test creation and execution time increased. Since the structural technique presented the best results in coverage and number of test cases, it has a chance of revealing more faults than the other techniques, due to its different characteristics , but this should be further investigated.

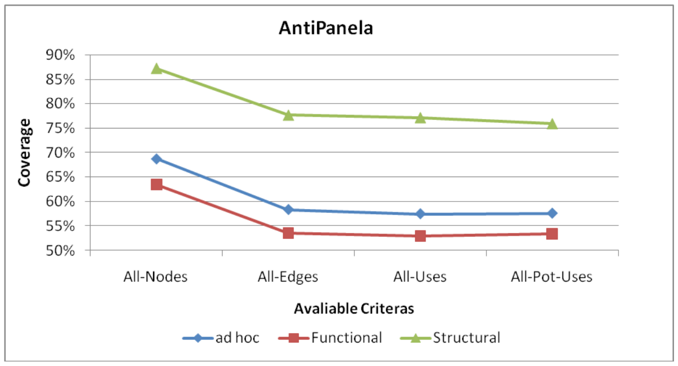


Figure 3. Coverage by program: AntiPanela

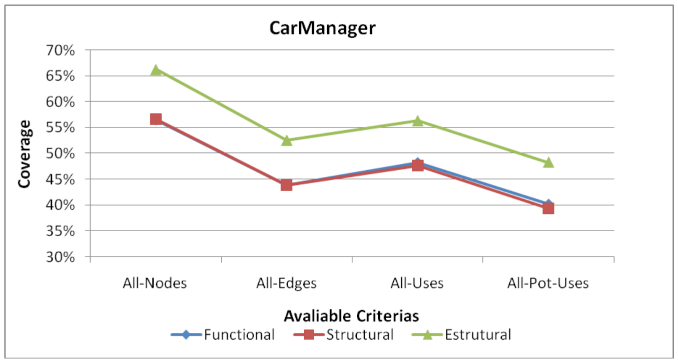


Figure 4. Coverage by program: CarManager

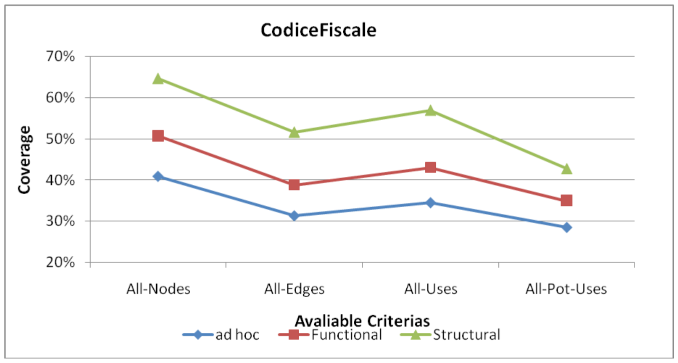


Figure 5. Coverage by program: CodiceFiscale

Thus, the more complex the program, such as CarManager and CodiceFiscale, the greater the time required to test it. In addition, the structural and functional techniques with JaBUTi/ME were used in actual practice by the majority of participants for the first time. All this information can be found in Tables III and IV, and Figures 6 and 7.

TABLE III. AVERAGE OF COVERAGE (%), TEST CASES AND FAULTS BY PROGRAM

	AntiPanela			CarManager			CodiceFiscale		
Criteria/Technique	*ad-hoc*	Functional	Structrual	*ad-hoc*	Functional	Structrual	*ad-hoc*	Functional	Structrual
All-Nodes	68,76	63,50	87,25	56,56	56,63	66,27	40,88	50,71	64,69
All-Edges	58,35	53,50	77,67	43,89	43,84	52,53	31,29	38,79	51,69
All-Uses	57,47	52,89	77,17	48,22	47,63	56,33	34,47	43,00	57,00
All-Pot-Uses	57,59	53,33	75,92	40,17	39,32	48,27	28,47	34,93	42,81
Number of Test Cases	11,13	10,56	13,33	7,75	9,16	10,62	7,94	7,29	11,56
Faults Found	3,73	2,67	3,33	1,75	1,79	1,00	1,59	1,69	2,50

TABLE IV. STATISTICS OF COVERAGE (%), TEST CASES AND FAULTS BY TECHNIQUE

	ad-hoc			Funcional			Structrual		
Criteria/Technique	Av	SD	Median	Av	SD	Median	Av	SD	Median
All-Nodes	55	20	56	57	18	61	72	18	73
All-Edges	45	18	44	46	17	50	59	18	59
All-Uses	47	19	48	48	17	50	62	18	63
All-Pot-Uses	42	19	40	43	18	44	54	20	54
Number of Test Cases	8,9	6,6	8,0	9,1	5,0	8,0	11,8	5,4	11,0
Faults Found	2,3	2,0	2,0	2,1	1,5	2,0	2,3	1,8	2,0

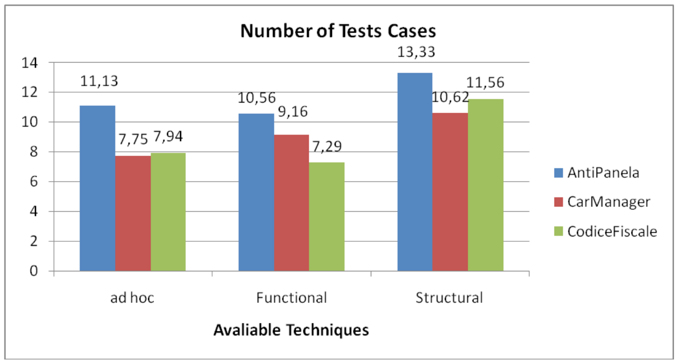


Figure 6. Number of test cases by program

Some data were lost during the experiment. The most common causes were: a) the participant did not save Form 2 – Test Case files correctly and was unable to send them to the course organizers; b) the participant did not initialize the programs correctly. This made it impossible to capture coverage information. Information loss reached about 20% for AntiPanela, 12% for CarManager, and 20% for CodiceFiscale.

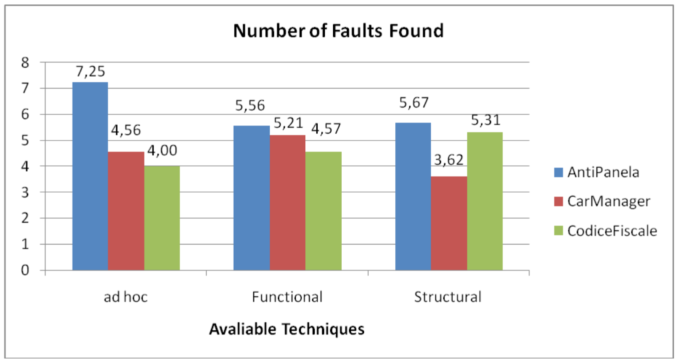


Figure 7. Number of faults found by program

Despite our emphasis on the importance of correctly following all the steps and executing the experiment, unfortunately deviations happen, the simultaneous supervision of around 20 participants per replication is very complex, and losses of data are inevitable. On Form 3 – Suggestions, 45% of the subjects asked for the presentation of other tools, including other languages, to give them more options for carrying out the tests. Thirty percent (30%) said that they would need more time to learn and practice the techniques. In other words, they assumed that they did not find more faults in the programs because of time constraints. Fifteen percent (15%) suggested not using Java ME programs.

In Form 4 – Course Evaluation, 100% of the participants said that the course had increased their knowledge of testing. Eighty-eight percent (88%) indicated that they felt confident in applying presented techniques. On the form, participants were asked to grade the level of knowledge acquired during the course. The average was 7.9 and the general grade for the course was 8.6, considering a 0 to 10 scale. Thus, the majority of participants approved and praised the initiative because testing techniques are not widely disseminated and it is difficult to find a free course on testing.

The participants made a number of comments about the course. The most important were: 1) that there is a lack of trained testing personnel; 2) that testing software is difficult; 3) that there is a shortage of testing tools. Many participants were interested in further studying JaBUTi/ME and in applying it in academic and professional programs.

The first step in the statistical analysis was to group the data by technique (*ad-hoc*, functional, and structural) rather than by program (AntiPanela, CarManager, and CodiceFiscale). The Shapiro-Wilk Test showed that the sample did not present a normal distribution. That is, it was necessary to use non-parametric statistical methods. The Kruskal-Wallis Test is robust for normality and its use makes it possible to check if there are relevant differences between the techniques evaluated in this paper. Its application showed that there are relevant differences between the three techniques for the criteria of coverage and number of test cases, as shown in Table V.

TABLE V. KRUSKAL-WALLIS TEST – RANK SUM TEST

Crit./Tech.	Ad-Hoc	Functional	Structural	p-value	Diff
All-Nodes	55,5	73,0	61,0	0,000019	Yes
Test Cases	8,0	11,0	8,0	0,009148	Yes
Faults	2,0	2,0	2,0	0,930200	Yes

Having discovered that there are differences between the techniques, it is necessary to find out what these differences are and then display the most effective technique for carrying out Java ME software tests on mobile devices. The Kruskal-Wallis Multiple Comparison Test is robust for normality and number of samples. It was used to compare pairs of techniques for each criterion. The result of this comparison is shown in Table VI.

TABLE VI. KRUSKAL-WALLIS MULTIPLE COMPARISON TEST

All-Nodes	Diff Observed	Crit Diff	Diff
ad-hoc X Structural	37,531753	20,86848	**Yes**
ad-hoc X Functional	3,86463	19,95248	No
Structural X Functional	33,667123	20,96088	**Yes**
Test Cases	Diff Observed	Crit Diff	Diff
ad-hoc X Structural	25,485814	20,69736	**Yes**
ad-hoc X Functional	5,804322	19,56142	No
Structural X Functional	19,681492	20,51164	No
Faults	Diff Observed	Crit Diff	Diff
ad-hoc X Structural	4,3252033	20,500141	No
ad-hoc X Functional	3,8666667	19,48089	No
Structural X Functional	0,4585366	20,31164	No

VI. CONCLUSION

Together, the three replications of the experiment highlight the importance and complexity of software testing in software engineering. All the different techniques and criteria focus on finding faults in types or parts of applications. The best known criteria include value limit analysis, equivalence partitioning, all-nodes, and all-edges.

Each technique has a particular focus, and techniques should be used together to find more faults in programs. The presented techniques help the tester select entry domain values systematically and may optimize the creation of test cases and increase fault detection.

The data collected in the replications of this experiment by Deus et al. (2008) show that the use of JaBUTi/ME and the structural technique help create test cases and consequently, provide greater coverage in mobile device programs. A statistical analysis showed that all techniques work equally well in detecting faults. In other words, the number of faults found using the evaluated techniques in this study did not differ significantly. However, it is important to point out that there are other characteristics besides fault detection that add value to software, which include the coverage of the software's internal structure, mainly important for program maintenance.

Thus, due to the techniques' similar performance, it is necessary to evaluate other criteria to choose the most efficient technique for ensuring mobile software product quality. Statistical analyses showed that among the evaluated techniques, there are significant differences in the criteria of code coverage and the number of test cases. Statistically, the structural technique performs better with respect to both of these aspects. More test cases were created and, consequently, greater coverage was achieved. Therefore, this initial study was not conclusive and should be replicated more times to increase its knowledge database.

Lessons were learned with each replication. This will help to improve the quality and objectivity of future studies that assess the results of experimentation packages.

Future research into mobile device software quality may include replication of this experimentation package using real mobile devices instead of emulators, creation of an effective method for mobile software quality control, and the evaluation of these or other techniques for conventional (non-mobile) software.

Smartphones are becoming more and more common and a large number of applications are created and freely distributed in different software repositories. Another option for future research is to use this package or to create a new package for Android environment that uses Java, that is a prerequisite for execution in JaBUTi/ME.

ACKNOWLEDGMENT

The authors would like to thank the Instituto de Informática – INF/UFG, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES – Brasil, and Fundação de Amparo à Pesquisa do Estado de Goiás – FAPEG – Brasil, which support this work.

REFERENCES

[1] IEEE, "IEEE standard glossary of software engineering terminology," International Standard, IEEE Computer Society Press, Standard 610.12-1990 (R2002), 2002.

[2] G1, "Mobile phones reach 256 milion of lines in july on brazil," Web page, Aug. 2012, [retrieved: Sep., 2013] (in Portuguese). [Online]. Available: http://g1.globo.com/tecnologia/noticia/2012/08/telefonia-movel-alcanca-256-milhoes-de-linhas-em-julho-no-brasil.html

[3] M. E. Delamaro, A. M. R. Vincenzi, and J. C. Maldonado, "A strategy to perform coverage testing of mobile applications," in *I International Workshop on Automation of Software Test – AST'2006*. New York, NY, USA: ACM Press, May 2006, pp. 118–124.

[4] P. Ammann and J. Offutt, *Introduction to Software Testing*. Cambridge University Press, 2008.

[5] N. Malevris, "On structurally testing Java programs effectively," in *Proceedings of the 3rd international symposium on Principles and practice of programming in Java*, ser. PPPJ'04. Trinity College Dublin, 2004, pp. 21–26, [retrieved: Sep., 2013]. [Online]. Available: http://dl.acm.org/citation.cfm?id=1071565.1071570

[6] P. Pocatilu, "Testing Java ME applications," *Informatica Economica*, vol. 12, no. 3, pp. 147–150, 2008.

[7] C. Hu and I. Neamtiu, "Automating gui testing for android applications," in *Proceedings of the 6th International Workshop on Automation of Software Test*, ser. AST'11. New York, NY, USA: ACM, 2011, pp. 77–83, [retrieved: Sep., 2013]. [Online]. Available: http://doi.acm.org/10.1145/1982595.1982612

[8] A. M. R. Vincenzi, W. E. Wong, M. E. Delamaro, and J. C. Maldonado, "JaBUTi: A coverage analysis tool for Java programs," in *XVII SBES – Brazilian Symposium on Software Engineering*. Manaus, AM, Brazil: Brazilian Computer Society (SBC), Oct. 2003, pp. 79–84.

[9] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in software engineering*. New York, NY, USA: Springer Heidelberg, 2012.

[10] G. E. P. Box, J. S. Hunter, and W. G. Hunter, *Statistics for Experimenters: Design, Innovation, and Discovery*, 2nd ed. Wiley-Interscience, May 2005.

[11] Y.-S. Ma, J. Offutt, and Y. R. Kwon, "Mujava: an automated class mutation system: Research articles," *STVR – Software Testing, Verification and Reliability*, vol. 15, no. 2, pp. 97–133, 2005.

How Exception Handling Constructions are Tested: An Initial Investigation with Open Source Software

Auri Marcelo Rizzo Vincenzi
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: auri@inf.ufg.br

João Carlos da Silva
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: jcs@inf.ufg.br

Plínio de Sá Leitão-Júnior
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: plinio@inf.ufg.br

José Carlos Maldonado
Inst. de Ciências Matemáticas e de Computação
Universidade de São Paulo, USP
São Carlos-SP, Brazil
e-mail: jcmaldon@icmc.usp.br

Márcio Eduardo Delamaro
Inst. de Ciências Matemáticas e de Computação
Universidade de São Paulo, USP
São Carlos-SP, Brazil
e-mail: delamaro@icmc.usp.br

Marcos Lordello Chaim
Escola de Artes, Ciências e Humanidades
Universidade de São Paulo, USP
São Paulo-SP, Brazil
e-mail: chaim@usp.br

***Abstract*—Software testing is one of the most important activities in software development to deliver quality to the final product. Aiming at high efficacy, high quality and a low-cost testing strategy, several testing techniques and criteria have been proposed in the last decades. In particular, structural testing techniques are among the most popular. The authors have extended traditional structural testing in order to meet this requirement, allowing its application to a software with exception handling structures to assess the coverage measurement of such structures. In this paper, we present control- and data-flow criteria to exercise such structures and then evaluate four well-known open source software projects according to these criteria. The results show that test cases for those software achieved low coverage of exception handling code and normal execution code as well. The work also shows that using test criteria which discriminate between exceptional and normal testing requirements might be useful to produce a better degree of information about the test set evaluated.**

***Keywords*-software engineering; testing criteria; structural testing; code coverage; testing tools**

I. INTRODUCTION

The exception handling mechanism available in a variety of languages brings improvements on how to deal with error handling or special conditions to product implementation. Instead of using the traditional return value for error indication, exceptions provide a more sophisticated approach for error handling. Despite its benefits, the use of exceptions brings additional challenges to system verification and validation.

By complementing other verification and validation techniques, like technical revision and formal methods, software testing enhances productivity and provides evidence of the reliability and the quality of the product. In addition, testing artifacts can be valuable information to other software engineering tasks, like debugging and maintenance.

Structural testing determines testing requirements from program source code. In general, structural testing criteria use a program representation known as def-use graph that abstracts the flow of control and variable usage of the program under testing. This paper describes a set of structural testing criteria for programming languages with exception handling mechanism. The underline control- and data-flow model is defined to represent such criteria and a tool which supports the model and implements the testing criteria instantiated for Java is described.

A set of Open Source Software (OSS) projects was evaluated in a large international project, aiming at encompass metric definition, measurement practices, data analysis, test suite definition, performance benchmarking, and indicator computation [1]. We applied structural testing to such projects to assess the quality of the available test sets. Some of these projects are employed in this paper to illustrate how OSS have been using exception handling constructions and how well their test sets exercise such structures.

The paper is organized as follows. In Section III, the exception handling mechanisms of Java language are described; Section IV presents the set of control- and data-flow based criteria we have extended to deal with exception handling constructions. In Section V, we present the data collected from four OSS projects, drawing a picture about the usage of exception handling constructions in those projects and how their OSS communities develop test cases for covering such pieces of code. In Section VI, we offer our conclusions and future work.

II. Related Work

Aberdour [2] compares close- and open-source software quality assurance and quality control, enumerating eleven differences. In the context of our work, four of them have a great impact since, according to Aberdour [2] in the OSS software development 1) the development methodology often is not defined or documented; 2) the testing and quality assurance methodology is unstructured and informal; 3) the defect discovery occurs from black-box testing late in the development process; and 4) the empirical evidences regarding quality are not collected. In one of the proposed guidelines to improve OSS development process, Aberdour [2] mentioned that the user based system testing should be complement with formal testing techniques and regression testing automation.

Considering specifically the testing process on OSS community, Zhao and Elbaum [3] conducted a survey with 200 OSS and found that instead of focusing on high quality milestone releases, the "release early, release often" process, traditionally adopted by the OSS community, results in a continual improvement by a large number of developers contributing iterations, enhancements, and corrections. With respect to the way OSS community test their software, Zhao and Elbaum [3] discovered that: 1) testing effort is concentrated on system testing; 2) fewer than 20% of OSS developers use test plans; 3) only 40% of projects use testing tools, but this percentage increases in case of Java, which has several available tools; 4) less than 50% of OSS use coverage concepts or tools to improve test quality.

A more recent study from Khanjani and Sulaiman [4] corroborates the ones above recognizing that despite the fact the open source development has seen remarkable success in recent years, there are a number of product quality issues and challenges facing the OSS development model. Considering exclusively the testing activities, they highlight the lack of knowledge of participants to understand the OSS system architecture and to create additional test cases for it.

Since we are interested to measure the coverage of exception handling code, we evaluated a few papers that discuss the analysis and testing of programs with exception handling structures. For instance, the works of Chatterjee et al. [5] and Choi et al. [6] present models to compute control- and data-flow information for dealing with exceptions, but no tool which implements the proposed models is available.

Sinha and Harrold [7] developed a family of criteria to deal with exception handling construction instantiated for the Java language. The testing criteria definition use a control- and a data-flow model known as interprocedural control-flow graph (ICFG) [8], [9], which is used for the identification of testing requirements. Java exceptions, as presented next, may be synchronized (explicitly raised by a `throw` statement) or unsynchronized (that can be raised at any time implicitly). The main limitation of ICFG is that, unlike our model, it does not represent unsynchronized exceptions. On the other hand, by considering only the synchronous exception it is possible to verify the type of exception to be raised when connecting nodes, so that no false edges are generated. To collect the data for the experiment, the authors used a tool named JABA, which is an acronym for Java Architecture for Bytecode Analysis. JABA provides language-dependent analysis for Java programs and is part of the Aristotle Analysis System [10], but JABA only performed the static analysis and, as soon as we know, there is no tool which implements such criteria.

III. Exception handling: features and requirements in Java

According to Perry *et al.* [11], a pervasive exception handling is required by almost anything that has an algorithmic flow, such as a design process, a workflow or a computer program. Exceptions are used not only as an implication of error, but also as an indication of deviations from the normal conditions established by the system. The main task of an exception handling mechanism in the context of programming languages is to overcome the problems posed by using the usual "return values from a function" as an indication of unexpected conditions. The use of exceptions to indicate error conditions ease the propagation of the erroneous state and also the implementation of the fault tolerance mechanism.

Programming languages like Java, C++ and Ada have similar exception handling mechanisms. In the case of Java, exceptions are represented by objects. We focus on Java for some reasons: 1) it is one of the largest used programming languages in this last decade [12]; 2) there are several open-source Java software with unit testing available; and 3) our previous effort on developing testing tools for Java [13].

Figure 1 shows part of the exception handling class hierarchy of the Java language. All those classes are part of the `java.lang` package. As Figure 1 shows, the `Throwable` class, an immediate subclass of Object, is the root class of the entire exception hierarchy. It has two direct subclasses: `Exception` and `Error`.

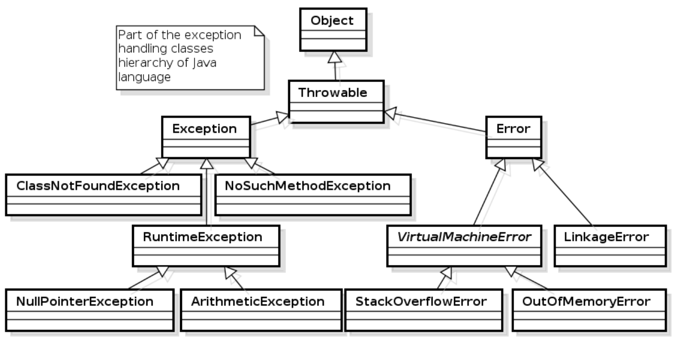


Figure 1. Part of the exception handling class hierarchy of Java [14].

Subclasses of `Exception` represent exceptional conditions that a normal Java program may handle and, except for `RuntimeException` and its subclasses, all the other subclasses of `Exception` are called "checked exceptions", i.e., exceptions that must be handled since they are verified at compilation time. `RuntimeException` and its subclasses, also known as "unchecked exceptions", represent runtime conditions that may generally occur in any Java method, but the method is not required to inform that it can raise runtime exceptions. Although they can be handled, unchecked exceptions are not identified at compilation time. On the other hand, all other standard exceptions a method can throw must be informed by means of a `throws` clause. A Java program should try to handle all standard exceptions, since they represent abnormal conditions that should be anticipated and caught to prevent program termination.

In addition to checked and unchecked exceptions, there are errors that can never be raised or handled since they are used to show serious problems with the Java virtual machine, the class loader or any other error which will abort program execution.

All checked exceptions may have exception handling code associated with them. This is done in Java by using a `try-catch-finally` construct. There are three possible valid combinations of these statements: `try-catch`, `try-catch-finally`, and `try-finally`. The `try` statement is composed by a `try` block. The `catch` block is composed by one or more `catch` clauses, responsible for specifying the exception handlers. The `catch` clause formal parameter determines the kind of exception it handles and the variable which will be assigned with the exception instance. The `finally` block, when present, is always executed, even in the presence of control-flow transfer statements like break, continue, and return in the body of the `try` block [14]. A feature of Java's exception handling mechanism is its non-resumable model, which means that once an exception is raised, the control flow returns to the first statement after the `try` statement responsible for handling such an exception.

In terms of testing, the exception handling mechanism affects the normal control-flow execution. Moreover, the set of instructions that may produce exceptions also has to be considered in the creation of basic CFG blocks. The set of instructions responsible for raising synchronous checked exceptions may be found elsewhere [15].

IV. Structural testing for exception handling

In this section, we present our approach to the structural testing of programs with exception handling constructs. It is part of a general framework that permits the application of control- and data-flow criteria to object oriented programs, in particular those developed using the Java language.

As part of this framework, a control- and a data-flow model were developed to accommodate our needs. The model is based on the analysis of bytecode programs instead of source code. This approach offers some advantages, it is language-independent and reflects the actual structure of a program under testing. The next subsections summarize our approach and the way it affects the testing of units with exception handling structures.

A. Control- and data-flow models

A common representation of the program under testing, known as Control-Flow Graph (CFG), is generally used to abstract the internal control flow of the tested unit. A program P can be decomposed in a set of disjoint blocks of statements so that the execution of the first statement inside a given block leads to the execution of all other statements in that block in the order they appear in. All statements in a block, except possibly the first, have a single predecessor. All statements in a block, except possibly the last, have exactly one successor. This means that there is no external control flow from/to statements in the middle of the block. In a CFG, such basic blocks are represented as vertex and the possible execution flow from one block to another is represented as directed edges. A CFG has a single entry node that represents the block which contains the entry instruction of the unit. An exit node has no outgoing edge.

A Def-use graph ($\mathcal{DUG}$) is an extension of the Control-Flow Graph including sets of variables defined and used on each CFG nodes [16]. Therefore, the $\mathcal{DUG}$ contains information about the data flow of the program under testing, characterizing associations between statements in which a definition occurs and statements in which a use is present.

It is out of the scope of this paper to discuss the complete OO testing approach and all the models and algorithms used to analyze the programs. It may be seen in [17].

Two points should be highlighted in the analysis of control-flow characteristics of a Java bytecode program:

- the use of intra-method subroutine calls. JVM has instructions `jsr`, `jsr_w` and `ret` that allow a piece of the method code to be "called" from several points in the same method. This is mostly used to implement the `finally` block of Java.
- exception handlers. Each piece of code inserted in a `catch` block of a Java program is an exception handler (EH). The execution of such a code is not performed by ordinary control-flow, but by the throwing of an exception. In the bytecode code the exception handler is not activated by ordinary instructions either. Each method has a table that describes where the handlers are located in the code and which piece of code they apply to. The flow of execution that is activated by an exception is represented in our $\mathcal{DUG}$ by a different type of edge, called an "exception edge".

To deal with Java's exception-handling mechanism, the underlying representation model, i.e., the $\mathcal{DUG}$, should reflect the control-flow during normal program execution and also during the occurrence of exceptions. To represent regular and exception control-flows, we use two kinds of edges: *regular edges* represent the regular control-flow, i.e., defined by the language statements; and *exception edges* represent the control-flow when an exception is raised. With such distinction, testing criteria can be defined to assess test coverage on normal execution flow and on exceptional execution flow.

B. Testing criteria

The basis to define testing criteria for exception handling structures is the concept of *exception-free path*:

An *exception-free path* is a path $\pi \mid \forall (n_i, n_j) \in \pi \Rightarrow (n_i, n_j)$ that is reachable through a path which does not contain any exception edge.

A path that includes a node n, which may only be reached through a path that contains an exception edge is an *exception-dependent* path.

To address explicitly the coverage of exception handlers code, two non-overlapping testing criteria were defined, so that the tester may concentrate on different aspects of a program at a time. Given the test set $T = \{t_1, t_2, ..., t_r\}$ and the corresponding set of paths $\Pi = \{\pi_1, \pi_2, ..., \pi_r\}$ executed by the elements of T, we define:

- *all-nodes-exception-independent* (All-Nodes$_{ei}$): Π satisfies the all-nodes-exception-independent criterion if every node $n \in N_{ei}$ is included in Π. In other words, this criterion requires that every node of the $\mathcal{DUG}$ graph, reachable through an exception-free path, is executed at least once.
- *all-nodes-exception-dependent* (All-Nodes$_{ed}$): Π satisfies the all-nodes-exception-dependent criterion if every node $n \in N_{ed}$ is included in Π. In other words, this criterion requires that every node of the $\mathcal{DUG}$ graph, not reachable through an exception-free path, is executed at least once.

Considering edges as testing requirements, we have:

- *all-edges-exception-independent* (All-Edges$_{ei}$): Π satisfies the all-edges-exception-independent criterion if every edge $e \in E_{ei}$ is included in Π. In other words, this criterion requires that every edge of the $\mathcal{DUG}$ graph that is reachable through an exception-free path is executed at least once.
- *all-edges-exception-dependent* (All-Edges$_{ed}$): Π satisfies the all-edges-exception-dependent criterion if every edge $e \in E_{ed}$ is included in Π. In other words, this criterion requires that every edge of the $\mathcal{DUG}$ graph not reachable through an exception-free path is executed at least once.

As with the all-nodes and all-edges criteria, we split the all-uses criterion [16], so that two sets of non-overlapping testing requirements are obtained. We named such criteria all-uses-exception-independent and all-uses-exception-dependent, respectively.

- *all-uses-exception-independent* (All-Uses$_{ei}$): Π satisfies the all-uses-exception-independent criterion if for every node $i \in N$ and for every variable $x \in def(i)$, Π includes a def-clear, exception-free path w.r.t. x from node i to every use of x. In other words, this criterion requires that every exception-independent def-c-use association (i, j, x) and every exception-independent def-p-use association $(i, (j, k), x)$ is exercised at least once for any given test case.
- *all-uses-exception-dependent* (All-Uses$_{ed}$): Π satisfies the all-uses-exception-dependent criterion if for every node $i \in N$ and for every variable $x \in def(i)$, Π includes a def-clear, exception-dependent path w.r.t. x from node i to every use of x. In other words, this criterion requires that every exception-dependent def-c-use association (i, j, x) and every exception-dependent def-p-use association $(i, (j, k), x)$ is exercised at least once for any given test case.

The use of testing criteria which consider exception codes when defining testing requirements can improve the testing activity by offering hints to the tester on how the code is organized, in terms of a “normal” or “abnormal” flow. Our test criteria may help in at least three situations:

- it is well known that much of the exception handling code is hard to test and it is left untested intentionally. With the indication of exception-dependent and exception-independent requirements, the tester may consider only the latter, with no need to analyze the feasibility of each, according to his/her goals;
- on the other hand, if the application requires the execution of an exception-dependent code, the use of our criteria can guide the tester indicating which requirements need an abnormal situation to be covered and suggesting a possible incremental testing strategy;
- exception dependent testing requirements can be used as a static code metric. For example, comparing the number of exception independent testing requirements against the number of exception independent requirements may give an indication of the cost of testing both normal and abnormal flow and, in some extent, of the complexity of these parts of the program. Other metrics like lines of code or cyclomatic complexity could also be used in this way if one considers these two types of code.

C. Automation aspects

To support the application of the structural testing criteria presented in the previous sections, we are working on the development of an Open Source testing tool called JaBUTi. We have worked on this tool since 2003 [13], improving its functionalities and extending its application to a variety of software products. Currently, besides testing Java programs at unit level, the tool may also be applied for unit testing of Aspect-Oriented programs [18], Java components [19], Java micro-edition, and mobile programs [20], among others. In addition, the tool can be easily employed to work with any language which generates bytecode as a result of the compilation process.

The steps for executing JaBUTi are depicted in Figure 2. The first step is the creation of a test session, which shows the classes that compose the program under testing and those we want to instrument for the collection of the execution trace. The second step is the generation of testing requirements by using the eight testing criteria. Then, it is necessary to instrument the selected classes. After instrumentation, the program under testing may be executed with one or more test cases and the coverage information is recorded. After test set execution, the covered requirements are identified and the current status of the test session is updated and visualized on testing reports with different levels of detail. With the reports the tester may decide whether to continue or stop the testing activity based on his/her previously defined stopping criterion.

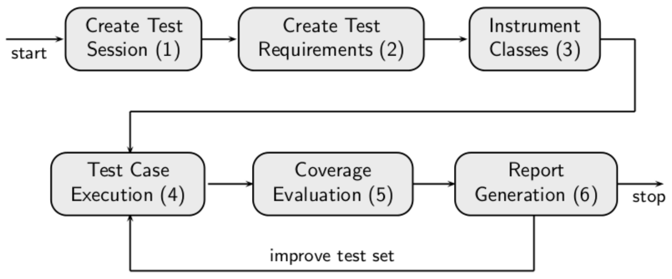


Figure 2. Steps of a test session execution.

We have successfully used JaBUTi on several projects and the tool has been released as an open source software to be used in the context of the QualiPSo project. The interested reader may consult [17], for further information.

V. Experimental application

In this section, we present the results obtained from the application of the exception-dependent criteria to a set of OSS. This initiative is part of our objectives in an attempt to identify the usual behavior of the OSS community while developing test sets for OSS.

Our first task was to make a static evaluation of some open source projects, namely HSQLDB, JUnit, JMeter, PMD, Weka, ServiceMix, Talend Open Studio, SpagoBI, Cimero, Jboss Application Server, Mondrian, Pentaho, and Spago. We have concluded that all of them have test sets associated with them and, as they are integrated with automated tools (Ant or Maven), it can be assumed that they are often run. However, despite this testing culture, the testing techniques applied by the OSS development community could not be identified with accuracy. Considering the current state of testing carried out by OSS communities, it can be observed that:

- in general, the only testing criterion applied is functional. There is no clear evidence of structural (control, data-flow) or fault based testing;
- there is no clear distinction between unit, integration, and system testing. Although there are test suites integrated into the build process (most projects use Ant or Maven to manage software compilation and packaging), there are no clearly defined test plans and strategies after the execution of the test suite. For example, how to proceed when failed test cases are found (e.g., if more than 10% of the test cases failed, the developers must be notified and the software package cannot be released).

A question which regards these test suites is: “Are *ad hoc* test suites sufficient to assign trustworthiness to OSS?” To answer this question we use an approach which comprises structural testing criteria for test set evaluation.

Formal standards like DO-178B [21] and ANSI/IEEE 1008-1987 [22] demand 100% statement and branch coverage for safety critical systems. Regardless of the level of coverage obtained, the importance of coverage testing does not lie on identifying which parts of the product were exercised during test set execution, but on identifying the ones which have not yet been executed.

Cornett [23] discusses the minimum acceptable code coverage and argues that a coverage level between 70-80% is a reasonable goal for system testing for the majority of software products. Moreover, Cornett [23] also defends that unit, integration, and system testing levels demand a decreasing coverage level since, in general, it is easier to achieve a higher coverage of a single unit than that of an entire system. An important point that has not been mentioned is how exception handling structures affect coverage level. It is not clear whether the 70-80% mentioned by Cornett considers normal and exception handling codes or only normal code. By using the testing criteria presented in Section IV, a more precise assessment of code coverage may be obtained.

As an initial investigation, we analyzed four traditional OSS: HSQLDB (version 1.9 Alpha 2), JMeter (version 2.3.2), JUnit (version 4.6), and PMD (version 5.0). The evaluation is performed via a testing tool that implements all the mentioned criteria, but we concentrate the analysis on the exception dependent ones. In this way, the restriction imposed by the selection of a OSS is the need that its unit test set run successfully, enabling the coverage information to be collected.

The OSS are implemented in Java and correspond to the last release available at the time the data was collected. We concentrate our effort on evaluating the impact of exception handling in these projects and how test sets were developed in order to cover exception code.

Our first evaluation consisted in identifying the size of the projects and the number of methods which employ exception handling constructions. The smallest OSS analyzed (JUnit) has 2,614 lines of code (LOC), and the biggest (HSQLDB) has 63,592 LOC. On average, at method level, the use of exception handlers construction is present on 8% of the total number of methods, percentage close to the average obtained by Sinha and Harrold [9] for a different set of programs. After performing the static analysis, we started the dynamic analysis.

We created an instrumented version of the programs under testing and executed the available test set against those versions, so that dynamic trace information could be collected and confronted with the structural testing criteria. Tables I, II, and III show the data obtained.

Tables I and II show the coverage after the execution of all available test sets developed by the OSS community for each program, considering the exception-independent and the exception-dependent testing criteria, respectively. For instance, the JMeter test set was the one which determined the highest coverage with respect to all testing criteria. For All-Nodes$_{ei}$, the test set covered 7,845 out of 20,462 required elements, 38.34% of coverage. As for the other testing criteria with higher complexity, the coverage percentage of the required elements were 28.27%, 26.55%, and 25.75%. In general, a level of coverage below 40% for these programs is very low

TABLE I. REQUIREMENT COVERAGE: EXCEPTION-INDEPENDENT CRITERIA

OSS	Criterion			
	All-Nodes$_{ei}$	**All-Edges$_{ei}$**	**All-Uses$_{ei}$**	**All-Pot-Uses$_{ei}$**
	Coverage (%)	**Coverage (%)**	**Coverage (%)**	**Coverage (%)**
HSQLDB	8,029 / 40,703 (19.73%)	7,476 / 45,098 (16.58%)	19,720 / 126,246 (15.62%)	67,847 / 458,843 (14.79%)
JMeter	7,845 / 20,462 (38.34%)	5,461 / 19,317 (28.27%)	10,935 / 41,180 (26.55%)	33,615 / 130,547(25.75%)
JUnit	608 / 1,951 (31.16%)	380 / 1,436 (26.46%)	631 / 2,624 (24.05%)	1,475 / 6,243 (23.63%)
PMD	7,938 / 21,184 (37.47%)	6,858 / 23,249 (29.50%)	13,331 / 57,552 (23.16%)	38,404 / 252,261 (15.22%)

TABLE II. REQUIREMENT COVERAGE: EXCEPTION-DEPENDENT CRITERIA

OSS	Criterion			
	All-Nodes$_{ed}$	**All-Edges$_{ed}$**	**All-Uses$_{ed}$**	**All-Pot-Uses$_{ed}$**
	Coverage (%)	**Coverage (%)**	**Coverage (%)**	**Coverage (%)**
HSQLDB	141 / 1,942 (7.26%)	49 / 6,513 (0.75%)	256 / 2,750 (9.31%)	3,591 / 38,032 (9,44%)
JMeter	51 / 1,541 (3.31%)	39 / 4,863 (0.80%)	52 / 2,093 (2.48%)	276 / 15,301 (1.80%)
JUnit	12 / 156 (7.69%)	9 / 184 (4.89%)	13 / 183 (7.10%)	29 / 632 (4.59%)
PMD	325 / 2,039 (15.94%)	121 / 3,814 (3.17%)	388 / 3590 (10.81%)	1689 / 20285 (8.33%)

TABLE III. EXCEPTION HANDLERS DATA AT METHOD LEVEL: ALL-NODES$_{ed}$ CRITERION

OSS	Number of methods	Number of requirements	Average	Number of methods with no coverage	Total coverage
HSQLDB	683	1,942	2.84%	669 (97.95%)	7.26%
JMeter	625	1,541	2.47%	595 (95.20)%	3.31%
JUnit	63	156	2.48%	57 (90.48%)	7.69%
PMD	374	2,039	5.45%	299 (79.95%)	15.94%

and demonstrates that much of the code is only executed by the users and that their test cases are probably not integrated in the official test set. In the case of HSQLDB, the percentage of coverage of the All-Nodes$_{ei}$ criterion is 19.73%, which means that more than 80% of source code is not executed by any official test case in the test set.

Table II shows the coverage obtained with respect to the exception-dependent criteria, i.e., those criteria which demand an exception to be raised for covering the testing requirements. Considering the most basic structural testing criterion (All-Nodes$_{ed}$), the highest coverage was determined by the test set of the PMD project, which executed 325 out of 2,039 testing requirements (15.94%). This is a clearly very low coverage and additional test sets should be developed at least to confirm that most of the exception handling construction in the program could be executed at least once.

When comparing such a coverage against the exception-independent criteria (Table I), one can see that even for the All-Nodes$_{ed}$ criterion, the level of coverage for all programs ranges from 19.73% for HSQLDB and 38.34% for JMeter. This implies that the provided test set for such programs has a very low coverage in terms of structural testing criteria, even for the criteria not related with exception handling.

In Table III, we present more detailed information about the total number of methods with exception handlers, the total number of testing requirements generated by the All-Nodes$_{ed}$ criterion, the average number of requirements per method, the number of methods which do not have exception handler construction executed by any test case, and the total coverage obtained for such a criterion. As Table III shows, there is a high percentage of methods with zero coverage against any exception-dependent criterion. For three programs, more than 90% of their methods have no test case to execute their exception handling constructions. The best program is PMD, for which the current test set is able to exercise 75 (20.05%) out of 374 methods with exception handlers, but still 79.95% of the methods are not executed by any test case.

Another point that might be inferred from Table III is that the exception handlers have normally few nodes, i.e., they are less complex in terms of logical structure. In fact, by analyzing such products, it is possible to observe that the majority of exception handlers have empty `catch` blocks, just avoiding the exception propagation but with no corrective action associated with it. The most complex exception handlers are found in PMD, which has on average 5.45 requirements per method, followed by HSQLDB with 2.84 requirements per method, considering the All-Nodes$_{ed}$ criterion.

These numbers show that all the analyzed projects reveal a low level of code coverage for code unrelated to exception handling structures. This is disturbing because it reveals the lack of concern from OSS communities on constructing a reference test set for their products. The tests are in fact performed *ad hoc* by the user and test cases are not incorporated in the official test set.

For exception handling criteria the situation is even worse. Although the complexity of exception handlers is not high – as shown by the number of testing requirements – the coverage of such testing requirements is very low. Many of the methods with this kind of code are not even executed once. In addition, there is no indication of test cases specifically designed to address exception handling.

In this scenario, the testing criteria presented in this paper may be of great help for developers, as they guide the tester through the process of selecting test cases that are or are not related to exception handling. Even if the adopted policy is not to execute exception handlers because they may be difficult to reach, our approach reveals which requirements could be neglected and which should be covered.

VI. Conclusions and future work

To support the control- and data-flow model and the defined testing criteria, we implemented a tool and presented experimental data collected from a set of four OSS. The experiment intended to assess the adequacy of pre-existent test sets against the set of exception-dependent structural testing criteria.

Our observations reveal that, for all the evaluated projects, the coverage of exception handling constructions was considerable low. For instance, the maximum coverage of the All-Nodes$_{ed}$ criterion was below 16%, which shows that, in general, there is no concern for the development of test cases to exercise exceptional conditions in the projects. Moreover, many exception constructions have empty `catch` blocks, which reveals that the exception handler, though present, is used only to avoid the spread of the exception, not to recover from an erroneous condition.

Even when evaluating the quality of the pre-existent test sets against the exception-independent criteria, the maximum coverage for the All-Nodes$_{ei}$ criterion was below 39%, which is generally regarded as a low level of coverage and an indicator that the test set should be improved. New versions of the analyzed software products may include additional test cases to improve the coverage with respect to the proposed testing criteria. This is an issue to be investigated; however, what this initial investigation indicates is that the open-source community should pursue more thorough test suites, especially addressing exception related code.

In future, we will continue to evaluate other OSS projects. Our aim is to finalize the evaluations of the previously developed test sets, to improve some of them based on the coverage criteria, to identify the contribution of the new added test cases in terms of their fault detection capability – considering the recorded faults in the bug tracker systems of these projects – and, finally, to define an incremental approach for testing OSS so that a minimal trustworthiness might be determined.

Acknowledgment

The authors would like to thank the Instituto de Informática – INF/UFG, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES – Brasil, and Fundação de Amparo à Pesquisa do Estado de Goiás – FAPEG – Brasil, which support this work.

References

[1] QualiPSo, "Qualipso project (quality platform for open source software)," Project Homepage – Europe Comission – Grant Number IST-FP6-IP-034763, 2007, [retrieved: Oct., 2013]. [Online]. Available: http://www.qualipso.org/

[2] M. Aberdour, "Achieving quality in open source software," *IEEE Software*, vol. 24, no. 1, pp. 58–64, 2007.

[3] L. Zhao and S. Elbaum, "A survey on quality related activities in open source," *SIGSOFT Softw. Eng. Notes*, vol. 25, no. 3, pp. 54–57, May 2000, [retrieved: Oct., 2013]. [Online]. Available: http://doi.acm.org/10.1145/505863.505878

[4] A. Khanjani and R. Sulaiman, "The process of quality assurance under open source software development," in *Computers Informatics (ISCI), 2011 IEEE Symposium on*, 2011, pp. 548–552.

[5] R. K. Chatterjee, B. G. Ryder, and W. A. Landi, "Complexity of concrete type-inference in the presence of exceptions," in *Lecture Notes in Computer Science*, vol. 1381. Springer, Apr. 1998, pp. 57–74.

[6] J.-D. Choi, D. Grove, M. Hind, and V. Sarkar, "Efficient and precise modeling of exceptions for the analysis of java programs," *SIGSOFT Software Engeneering Notes*, vol. 24, no. 5, pp. 21–31, 1999.

[7] S. Sinha and M. J. Harrold, "Criteria for testing exception-handling constructs in Java programs," in *International Conference on Software Maintenance*. Oxford, England: IEEE Computer Society Press, Aug. 1999, pp. 265–274.

[8] ——, "Analysis of programs with exception-handling constructs," in *ICSM'98 – International Conference on Software Maintenance*, Bethesda, MD, Nov. 1998, pp. 348–357.

[9] ——, "Analysis and testing of programs with exception-handling constructs," *IEEE Transactions on Software Engineering*, vol. 26, no. 9, pp. 849–871, Sep. 2000.

[10] M. J. Harrold, L. Larsen, J. Lloyd, D. Nedved, M. Page, G. Rothermel, M. Singh, and M. Smith, "Aristotle: a system for development of program analysis based tools," in *ACM-SE 33: Proceedings of the 33rd annual on Southeast regional conference*. New York, NY, USA: ACM, 1995, pp. 110–119.

[11] D. E. Perry, A. Romanovsky, and A. Tripathi, "Current trends in exception handling," *ieeese*, vol. 26, no. 10, pp. 921–922, Oct. 2000.

[12] TIOBE Software BV, "TIOBE Index," Web site, Sep. 2013, [retrieved: Oct., 2013]. [Online]. Available: http://www.tiobe.com/

[13] A. M. R. Vincenzi, W. E. Wong, M. E. Delamaro, and J. C. Maldonado, "JaBUTi: A coverage analysis tool for Java programs," in *XVII SBES – Brazilian Symposium on Software Engineering*. Manaus, AM, Brazil: Brazilian Computer Society (SBC), Oct. 2003, pp. 79–84.

[14] J. Gosling, B. Joy, G. Steele, and G. Bracha, *The Java Language Specification*, 3rd ed. Addison Wesley, Jun. 2005.

[15] T. Lindholm and F. Yellin, *The Java Virtual Machine Specification*, 2nd ed. Addison-Wesley, 1999.

[16] S. Rapps and E. J. Weyuker, "Selecting software test data using data flow information," *IEEE Transactions on Software Engineering*, vol. 11, no. 4, pp. 367–375, Apr. 1985.

[17] A. M. R. Vincenzi, M. E. Delamaro, W. E. Wong, and J. C. Maldonado, "Establishing structural testing criteria for Java bytecode," *Software Practice and Experience*, vol. 36, no. 14, pp. 1513–1541, Nov. 2006.

[18] O. A. L. Lemos, A. M. R. Vincenzi, J. C. Maldonado, and P. C. Masiero, "Control and data flow structural testing criteria for aspect-oriented programs," *The Journal of Systems and Software*, vol. 80, no. 6, pp. 862–882, Jun. 2007.

[19] A. M. R. Vincenzi, J. C. Maldonado, W. E. Wong, and M. E. Delamaro, "Coverage testing of Java programs and components," *Journal of Science of Computer Programming*, vol. 56, no. 1-2, pp. 211–230, Apr. 2005.

[20] M. E. Delamaro and A. M. R. Vincenzi, "Structural testing of mobile agents," in *III International Workshop on Scientific Engineering of Java Distributed Applications (FIDJI'2003)*, ser. Lecture Notes on Computer Science, E. A. Nicolas Guelfi and G. Reggio, Eds. Springer, Nov. 2003, pp. 73–85.

[21] RTCA/EUROCAE, "Software considerations in airborne systems and equipment certification," Radio Technical Commission for Aeronautics – RTCA & European Organization for Civil Aviation Equipment – EUROCAE, Washington, D.C., EUA, Relatóro Técnico DO-178B/ED12B, Dec. 1992.

[22] IEEE, "IEEE standard for software unit testing," IEEE Computer Society Press, Standard ANSI/IEEE Std 1008-1987, 1987.

[23] S. Cornett, "Minimum acceptable code coverage," On-line article, 2007, [retrieved: Oct., 2013]. [Online]. Available: http://www.bullseye.com/minimum.html

Towards Scalable Bug Localization using the Edit Distance of Call Traces

Themistoklis Diamantopoulos and Andreas Symeonidis
Information Technologies Institute, Centre for Research and Technology Hellas
Electrical and Computer Engineering Dept., Aristotle University of Thessaloniki
Thessaloniki, Greece
{thdiaman,asymeon}@iti.gr

***Abstract*—Locating software bugs is a difficult task, especially if they do not lead to crashes. Current research on automating non-crashing bug detection dictates collecting function call traces and representing them as graphs, and reducing the graphs before applying a subgraph mining algorithm. A ranking of potentially buggy functions is derived using frequency statistics for each node (function) in the correct and incorrect set of traces. Although most existing techniques are effective, they do not achieve scalability. To address this issue, this paper suggests reducing the graph dataset in order to isolate the graphs that are significant in localizing bugs. To this end, we propose the use of tree edit distance algorithms to identify the traces that are closer to each other, while belonging to different sets. The scalability of two proposed algorithms, an exact and a faster approximate one, is evaluated using a dataset derived from a real-world application. Finally, although the main scope of this work lies in scalability, the results indicate that there is no compromise in effectiveness.**

***Keywords*—automated debugging; dynamic bug detection; frequent subgraph mining; tree edit distance**

I. Introduction

Software reliability has grown to be a major concern for both academia and the industry. Software bugs lead to faulty software and dissatisfied customers, as testing and debugging are quite costly even compared to the development phase. As software grows more and more complex, though, identifying and eliminating software bugs has become a challenging task.

There are two types of bugs: *crashing* and *non-crashing* ones. The former lead to program crashes, thus they are easier to locate by tracing the call stack at the time of the crash. The latter are logic errors that do not lead to crashes and thus do not produce stack traces. Since dynamic analysis is performed to detect such bugs, the field is known as *dynamic bug detection*. The techniques may be classified according to the granularity of the source code instrumentation approach. Highly granular approaches involve inserting checks in different source code positions, either in the form of counters [1] or boolean predicates [2] while others involve inserting checks at block level [3], where blocks are fragments between branches. Counter-level and block-level approaches are quite precise in localizing bugs. However, since the rise of *Object Oriented Programming* and *Functional Programming* has led to preference for small comprehensive functions, instrumenting functions is effective, as long as proper programming paradigms are employed. Function-level approaches apply *Graph Mining* techniques to call traces to identify which subgraphs are more frequent in incorrect than in correct runs [4]–[6].

The steps used to localize bugs are common. The generated call traces constitute a dataset that has to be mined in order to detect bugs; and this is where the problems start. Even at function-level, datasets are usually huge. For a small application, with, e.g., 150 functions, there may be couples of thousands of transitions among them. In this context, creating an effective, yet also scalable, solution is a challenging problem. And, though it has been broadly studied, most literature approaches focus on reducing the size of each trace, without reducing the number of traces in the dataset.

In this paper, we present a novel approach towards highly scalable Graph Mining solutions for function-level traces. The main contribution lies in the problem formulation, the reduction of the call trace dataset size through different alternatives, and the construction of a realistic dataset to test upon. Dataset size reduction is confronted using tree edit distance algorithms, while the potential benefits and drawbacks with respect to different solutions are discussed. Furthermore, the applicability of several function-level dynamic bug detection techniques in real applications is discussed and the efficiency and effectiveness of our variations are evaluated against them.

Section II of the paper reviews current literature on function-level dynamic bug detection, illustrating the general procedure followed to mine the traces and identify the Graph Mining problems. Section III provides an overview of alternative solutions to known scalability issues. The construction of a realistic dataset that illustrates our contribution is explained in section IV. Finally, our implementation is evaluated in terms of efficiency and effectiveness in section V, while section VI concludes the paper and provides insight for further research.

II. Function-Level Dynamic Bug Detection

In this section, we discuss the steps of constructing a graph dataset, reducing it, and applying Graph Mining techniques to provide the ranking of possibly buggy functions.

A. Graph Dataset Construction

Given a set of tests, program functions are instrumented and the tests are run to produce a set of *call traces* S. A call trace is initially a rooted ordered tree, with the `main` function as its root. Two more sets, $S_{correct}$ and $S_{incorrect}$ are defined, corresponding to correct and incorrect executions, where correctness is determined by an oracle. Thus, the tree (or graph, since all trees are graphs) dataset is constructed.

B. Graph Reduction

Since graphs are large, with hundreds of nodes, applying any mining algorithm is inefficient. Thus, *graph reduction* is performed to reduce the size of each graph while keeping useful information. Figure 1 depicts reduction techniques.

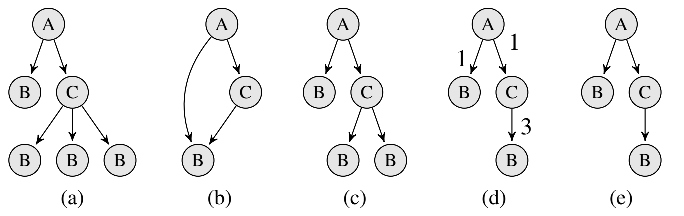


Fig. 1. An example call graph (a) and four different reduced graphs with respect to the reduction techniques, including (b) total reduction, (c) one-two-many reduction, (d) subtree reduction and (e) simple tree reduction.

The first technique, known as *total reduction*, is presented by Liu et al. [4]. The authors create a graph using each edge of the initial call graph once and discard any structural information (i.e., tree levels). Total reduction is the most efficient reduction method since it actually preserves minimum information.

However, since total reduction fails to capture the structure of the call graph, different alternatives have been applied to preserve more information, while keeping the graph as small as possible. A straightforward solution is the one proposed by Di Fatta et al. [5]; the authors perform *one-two-many* reduction, preserving tree structure by keeping two child nodes whenever the children of a node are more than two (see Figure 1c).

Eichinger et al. [6] claim that total reduction and one-two-many reduction are not sufficient, since they discard call frequency information. According to the authors, the number of times (i.e., frequency) that a function calls another function is crucial since it can capture bugs that may occur in, e.g., the third or fourth time the function is called. Thus, they propose *subtree reduction*, a technique that preserves both the structure of the tree and the frequency of function calls (see Figure 1d).

Reduction techniques are based on a compromise between information loss and scalability. Although subtree reduction maintains most information, it is quite inefficient since it adds a weight parameter to the graph. Since the scope of this work lies in scalability, we propose using a reduction technique, which we call *simple tree reduction*, shown in Figure 1e. Reducing a graph using simple tree reduction involves traversing the nodes once and deleting any duplicates as long as they are on the same level. The reduced graph is a satisfactory representation of the original one since large part of its structure is preserved.

C. Graph Mining

Upon reduction, the problem lies in determining the nodes that are frequent in the incorrect set $S_{incorrect}$ and infrequent in the correct set $S_{correct}$. Intuitively, if a function is called every time the result is incorrect, it is highly possible to have a bug. However, having more than one function with the same frequency is also possible. Thus, the Graph Mining algorithm should find the closed frequent subgraphs, i.e., the subgraphs for which no supergraph has greater support in $S_{incorrect}$.

Finding frequent subgraphs in a graph dataset, known as *Frequent Subgraph Mining (FSM)*, is a well-known problem. State-of-the-art algorithms include, e.g., gSpan [7]. Furthermore, since these graphs are actually trees, several *Frequent Subtree Mining (FTM)* algorithms, such as FreeTreeMiner [8], may be used as well. Although those algorithms are applicable to the problem, there is strong preference for *CloseGraph* [9], an algorithm that is highly scalable since it prunes unnecessary input and outputs only closed frequent subgraphs.

D. Ranking

The output of CloseGraph is a set of frequent subgraphs, along with their support in the correct and the incorrect set. Hence, the question is how can a ranking of possibly buggy functions be created by such a set. It is typical to use DM techniques based on *support* and *confidence* to determine the interesting subgraphs. For instance, Di Fatta et al. [5] suggest ranking the functions according to their support in the failing set. According to Eichinger et al. [6], this type of ranking can be called *structural* and for each function f is defined as:

$$P_s(f) = support(f, S_{incorrect}) \tag{1}$$

The support of each function in the failing set $S_{incorrect}$ provides a fairly effective ranking. However, the scoring is not sufficient, since it does not take confidence into account. Furthermore, finding the support only on incorrect executions yields skewed results, since a function with large support in both $S_{correct}$ and $S_{incorrect}$ would be ranked high, even though it may be insignificant with respect to the bug.

Several variations of the structural ranking have emerged in order to overcome the aforementioned issues [2][5]. In this paper, we use an entropy-based ranking technique proposed by Eichinger et al. [6] since it is proven to outperform the other techniques. The main intuition behind this ranking technique is to identify the edges that are most significant to discriminate between correct and incorrect call traces. A table is created with columns corresponding to subgraph edges and rows corresponding to graphs. The table holds the support of each edge in every graph. Consider the example of Table I:

TABLE I. ENTROPY-BASED RANKING EXAMPLE

$Graph$	$f_1 \to f_2$	$f_1 \to f_3$	$f_2 \to f_4$	$\dots$	$Class$
G_1	4	7	2	$\dots$	$correct$
G_2	9	5	8	$\dots$	$incorrect$
G_3	6	3	1	$\dots$	$correct$
$\dots$	$\dots$	$\dots$	$\dots$	$\dots$	$\dots$

where $F = f_1, f_2, \dots$ is the set of functions and $G = G_1, G_2, \dots$ is the set of graphs. Supposing subgraph SG_1 appears 4 times in graph G_1 and edge $f_1 \to f_2 \in SG_1$, the support of the edge in graph G_1 is 4. As one might observe, the problem is actually a *feature selection* problem, i.e., defining the features (edges) that discriminate between the values of the class feature ($correct$, $incorrect$). Thus, any feature selection algorithm may be used to determine the most significant features. Eichinger et al. [6] calculate the information gain for each feature, and interpret the result for each feature (ranging from 0 to 1) as the probability of it being responsible for a bug. The respective probability $P_e(f)$ for a node (function) is determined by the maximum probability of all the edges it is connected to.

The structural ranking P_s and the entropy-based ranking P_e are used to compute the combined ranking as follows:

$$P(f) = \frac{P_e(f)}{2 \max\limits_{f \in F} P_e(f)} + \frac{P_s(f)}{2 \max\limits_{f \in F} P_s(f)} \tag{2}$$

where the maximum values at the denominator are used in order to normalize the weighting of each ranking.

III. Reducing the Graph Dataset

The steps given in Section II are common for all function-level bug detection algorithms. Several researchers have indicated the need for scalability, which is generally accomplished by reducing the graphs (see subsection II-B). Ideally, the useful information of the graph is retained while its size is minimized. However, even upon reduction, the number of graphs in the dataset is large, thus making the mining step quite inefficient.

Although a dataset of several graphs is given, not all of them are equally useful in locating the bug. Consider a scenario for the `grep` program. Assume the program has a bug that results in faulty executions when the `?` character is used in a *Regular Expression (RE)*, such that the appropriate words are not returned, if the preceding element appears 0 times. Normally, if a symbol is succeeded by the `?` character, then it may be found 0 or 1 times exactly. Consider running the `grep` program for one word at a time for the following phrase:

```
there once was a cat that ate a rat
  and then it sat on a yellow mat
```

In this text, the RE `[a-z]*c?at` should match the words in the set $S_{matched} = \{$`cat`, `that`, `rat`, `sat`, `mat`$\}$, i.e., all words having any letter from `a` to `z` 0 or more times, followed by the letter `c` 0 or 1 times exactly, and followed by letters `a` and `t`. Instead it only matches the word `cat`. Consider also the set of words that are not matched $S_{unmatched} = \{$`there`, `once`, `was`, `a`$_{(1)}$, `ate`, `a`$_{(2)}$, `and`, `then`, `it`, `on`, `a`$_{(3)}$, `yellow`$\}$. Assuming that all the possible traces are created, several of them, such as the ones created from the $S_{matched}$ set, are actually much more significant in identifying the bug, since it actually resides only on the $S_{matched}$ set. Thus, traces of `cat` and `rat` should be more *similar* than traces of `cat` and `yellow`. In fact, when executing the `cat` and `rat` scenarios, many function calls coincide. This is however also true for traces of `was` and `it`. Intuitively, determining which traces are highly indicative of the bug can be based on the similarity between them as well as whether they are correct or incorrect. Thus, correct executions that are *similar* to the incorrect ones (e.g., `rat` may be close to `cat`) should isolate more easily the buggy functions. On the other hand, when two correct (or incorrect) executions are quite close to each other (e.g., the traces from `was` and `it` could be quite similar), then one of them should provide all necessary information.

The example is formed such that it is easy to understand. One could ask why not select test cases by hand, so that they are discriminating. However, this is usually impossible since real scenarios are much more complex, e.g., for the `grep` case there may be passages instead of words. In addition, certain executions may seem similar, yet be significantly different with respect to the call traces. Thus, there is the need for a similarity metric between two traces. Having such a metric, one can apply the call trace selector algorithm shown in Figure 2.

As shown in this figure, the algorithm requires as input the correct and incorrect sets, $S_{correct}$ and $S_{incorrect}$, along with parameter n, which controls how many graphs are going to be retained per set. Initially, the set D, which contains all correct-incorrect pairs of graphs, is sorted according to the similarity of each pair. The set $S'_{correct}$ contains the first n correct unique graphs that are found in the sorted set D, i.e., the n correct

```
Input: $n$, $S_{correct}$, $S_{incorrect}$
Output: $S'_{correct}$, $S'_{incorrect}$

$D = \{(g_1, g_2)\ \forall g_1 \in S_{correct}, g_2 \in S_{incorrect}\}$
sort($D$, key=similarity($g_1, g_2$))
$S'_{correct}$ =First($n, \{g_1 :\ g_1 \in d \in D\}$)
$S'_{incorrect}$ =First($n, \{g_2 :\ g_2 \in d \in D\}$)
```

Fig. 2. The call trace selector algorithm that receives the two sets of graphs as input (correct and incorrect) and its output is two new subsets of them.

graphs that belong to the most similar pairs d of D. The set $S'_{incorrect}$ contains the first n incorrect unique graphs that are found in the sorted set D. For example, given $n = 2$ and $D = \{d_1, d_2, d_3\} = \{(g_1, g_3), (g_1, g_4), (g_2, g_5)\}$ so that the similarity of pair d_1 is larger than that of d_2 and the similarity of d_2 is larger than that of d_3, the sets $S'_{correct}$ and $S'_{incorrect}$ are $\{g_1, g_2\}$ and $\{g_3, g_4\}$ respectively. Function `sort` sorts the set according to the `key` and `index` provides the index of an element. Thus, the issue is how to determine similarity between two graphs, i.e., how to implement the function `similarity`.

A metric widely used to represent the similarity between two strings is the *String Edit Distance (SED)*. SED is defined as the number of edit operations required to transform one string to the other. SED operations usually contain insertion or deletion of characters. Concerning trees, such as the ones of our dataset, *Tree Edit Distance (TED)* algorithms can be used to calculate the distance between two of them. The following subsections provide a definition of the TED problem and two well known algorithms of current literature in finding TED.

A. The Tree Edit Distance Problem

The TED problem was originally posed by Tai [10] in 1979. The possible *edit operations* are defined in Figure 3.

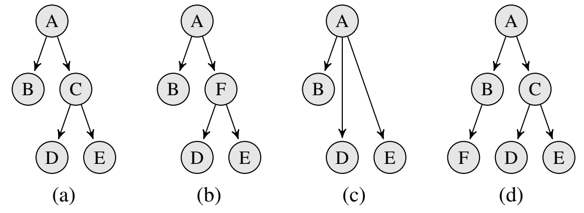


Fig. 3. An example tree (a) and three different edit operations: (b) node relabeling, (c) node deletion, and (d) node insertion.

Node relabeling concerns simply changing the label of a node (see Figure 3b). *Node deletion* is performed by deleting a node of the tree and reassigning any children it had so that they become children of the deleted node's parent. For example in Figure 3c, the children of deleted node `C` are reassigned to `C`'s parent `A`. Finally, *node insertion* concerns inserting a new node in a position in the tree, such as inserting node `F` in Figure 3d. Assuming a *cost function* is defined for each operation, an *edit script* between two trees T_1, T_2 is a sequence of operations required to turn T_1 into T_2, and its cost is the aggregate cost of them. Thus, the TED problem is defined as determining the *optimal edit script*, i.e., the one with the minimum cost.

B. Zhang-Shasha Algorithm

Let $\delta(T_1, T_2)$ be the edit distance between trees T_1 and T_2, and $\gamma(l_1 \rightarrow l_2)$ be the cost of the edit operation from l_1 to l_2.

A simple algorithm for computing TED is defined as follows:

$$\delta(\theta,\theta) = 0 \tag{3}$$
$$\delta(T_1,\theta) = \delta(T_1 - u,\theta) + \gamma(u \to \lambda) \tag{4}$$
$$\delta(\theta,T_2) = \delta(\theta,T_2 - v) + \gamma(\lambda \to v) \tag{5}$$
$$\delta(T_1,T_2) = min \begin{cases} \delta(T_1 - u, T_2) + \gamma(u \to \lambda) \\ \delta(T_1, T_2 - v) + \gamma(\lambda \to v) \\ \delta(T_1(u), T_2(v)) + \delta(T_1 - T_1(u), \\ \qquad T_2 - T_2(v)) + \gamma(\lambda \to v) \end{cases} \tag{6}$$

where $T - u$ denotes tree T without node u and $T - T(u)$ denotes tree T without u or any of each children. Parameter λ is the performed edit operation. The *Zhang-Shasha* algorithm, which was named after its authors, K. Zhang and D. Shasha [11], uses *Dynamic Programming (DP)* in order to compute the TED. The *keyroots* of a tree T are defined as:

$$\textit{keyroots}(T) = \{root(T)\} \cup \{u \in T : \text{u has left siblings}\} \tag{7}$$

Given (7), the *relevant subtrees* of T are defined as:

$$\textit{relevant\_subtrees}(T) = \bigcup_u \{T(u)\},\ \forall u \in keyroots(T) \tag{8}$$

Thus, the algorithm recursively computes the TED by finding the relevant subtrees and applying equations (3)–(6).

C. pq-Grams Algorithm

Several algorithms solve the TED problem. However, even the most efficient ones lack scalability, since the polynomial order of the problem is high. A promising way of reducing complexity is by approximating the TED instead of computing its exact value. Approximate TED algorithms can generally be effective enough when results do not need to be exact. In the call trace scenario, the TED is a value denoting the similarity of two trees, thus, even if it is approximate, it shall provide with the appropriate n most significant graphs as in Figure 2.

Such an approximate TED algorithm is the *pq-Grams* based algorithm proposed by Augsten et al. [12]. The authors define *pq*-Grams as a port of known string *q*-grams to trees. An example tree and its *pq*-Grams are shown in Figure 4. The p and q parameters define the *stem* and the *base* of the *pq*-Gram, respectively. Let $p = 2$ and $q = 3$, the stem of the first *pq*-Gram of Figure 4c is $\{*, \texttt{A}\}$ and its base is $\{*, *, \texttt{B}\}$. Since the *pq*-Grams for the tree of Figure 4a cannot be directly created, an intermediate step of extending the tree with dummy nodes is shown in Figure 4b. The *pq-Gram profile* is the set of all *pq*-Grams of a tree (see Figure 4c), while the *pq-Gram index* of the tree is defined as the bag of all label tuples for the tree. The *pq*-Gram index for the tree of Figure 4 is:

$$I(T) = \{*A{*}{*}B, *A{*}BC, *ABC*, *AC{*}{*}, AB{*}{*}{*}, \\ AC{*}{*}D, AC{*}DE, ACDE*, ACE{*}{*}, CD{*}{*}{*}, CE{*}{*}{*}\} \tag{9}$$

According to Augsten et al. [12], the TED between two trees is effectively approximated by the distance between their *pq*-Gram indexes. Let $I(T)$ be the *pq*-Gram index of tree T, the *pq*-Gram distance between trees T_1 and T_2 is defined as:

$$\delta(T_1,T_2) = |I(T_1) \cup I(T_2)| - 2|I(T_1) \cap I(T_2)| \tag{10}$$

Equation (10) provides a fast way of approximating the TED between any pair of trees of the dataset. Thus, the *pq*-Gram distance function can be used in place of the `similarity` function which is required by the algorithm shown in Figure 2.

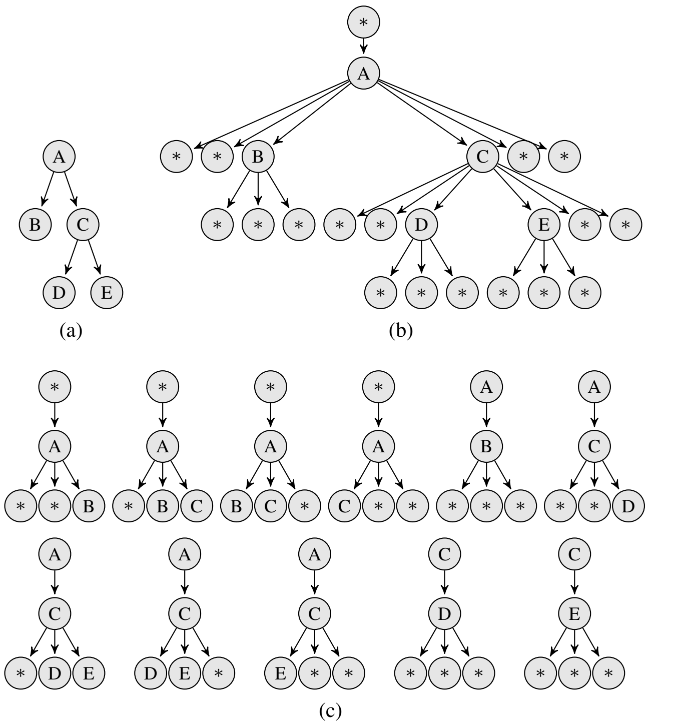


Fig. 4. A pq-Grams example for $p = 2$ and $q = 3$, containing (a) an example tree, (b) its extended form for $p = 2$ and $q = 3$, and (c) its pq-Grams.

IV. Dataset

The techniques of section II are effective for bug localization in small applications. For example, Eichinger et al. [6] evaluate their method against two known literature bug localization techniques ([4] and [5]) using a small dataset. Although effectiveness is irrefutable, efficiency is not thoroughly tested since the dataset is too small to resemble a real application. Indicatively, the size of the program is almost 2 pages of code, leading to graphs of roughly 20 nodes after the reduction step.

Since the main scope of this paper lies in achieving scalability in order to locate bugs of real applications, a larger dataset has to be used. The dataset was generated using the source code of `daisydiff` [13], a `Java` application that compares `html` files. We used the 1.2 version of `daisydiff` and planted 3 types of bugs in the code, as shown in Table II.

TABLE II. Planted Bugs

Bugs	Description	Function Calls
1	Wrong limit conditions (Forgot `+1`)	17509
2	Missing condition (Forgot a $<$ check)	54137
3	Wrong condition ($>$ instead of $<$)	78837

These bugs do not aim to cover possible bug classes, as in [6], rather to test algorithm efficiency. Three scenarios with different number of function calls are created to demonstrate our proof of concept. The bug-free and the three buggy versions were run 100 times given different inputs. The application has almost 70 files with 9500 lines, leading to graphs of almost 750 nodes after reduction. The dataset is given online in [14].

TABLE III. AVERAGE ELAPSED TIME (IN SECONDS) FOR THE DIFFERENT PHASES OF THE ALGORITHMS

	pq-Grams							NoTED	ZhangShasha						
	5	10	15	20	25	30	35	-	5	10	15	20	25	30	35
Graph Parsing	7.81	7.15	7.11	7.15	7.13	7.12	7.13	7.14	8.71	7.05	7.06	7.02	7.03	7.02	7.02
Graph Reduction	4.03	3.97	4.00	4.04	4.03	3.96	3.97	3.93	4.07	3.98	3.97	3.94	3.99	3.92	3.94
Dataset Reduction	84.22	84.10	84.91	83.99	83.94	83.86	84.07	0.00	188.23	187.90	187.37	187.35	187.81	187.41	187.28
Subgraph Mining	7.55	27.10	54.69	131.63	440.51	412.46	450.87	4712.54	5.87	40.05	69.91	149.03	389.21	436.68	611.15
Ranking Calculation	0.56	2.25	6.51	16.62	34.82	36.47	45.86	533.59	0.58	2.77	7.23	16.33	33.87	38.31	55.12
Total	104.17	124.57	157.22	243.43	570.43	543.87	591.90	5257.20	207.46	241.75	275.54	363.67	621.91	673.34	864.51

TABLE IV. RANKING POSITION AND PERCENTAGE OF FUNCTIONS TO BE EXAMINED TO FIND THE BUGS

		pq-Grams							NoTED	ZhangShasha						
		5	10	15	20	25	30	35	-	5	10	15	20	25	30	35
Bug 1	Position	7	7	31	9	8	8	8	8	7	6	9	8	9	8	8
	Percentage	1.10%	1.10%	4.87%	1.41%	1.26%	1.26%	1.26%	1.26%	1.10%	0.94%	1.41%	1.26%	1.41%	1.26%	1.26%
Bug 2	Position	5	5	9	9	9	10	9	9	5	5	9	9	9	9	9
	Percentage	0.68%	0.68%	1.22%	1.22%	1.22%	1.36%	1.22%	1.22%	0.68%	0.68%	1.22%	1.22%	1.22%	1.22%	1.22%
Bug 3	Position	105	105	341	27	3	1	15	17	105	337	342	352	1	1	1
	Percentage	13.51%	13.51%	43.89%	3.47%	0.39%	0.13%	1.93%	2.19%	13.51%	43.37%	44.02%	45.30%	0.13%	0.13%	0.13%

V. EVALUATION

This section presents the results of applying three different algorithms to the dataset described in section IV.

A. Experimental Setup

We implemented three algorithms to test the validity of our dataset reduction hypothesis. The first is the algorithm by Eichinger et al. [6] as explained in section II. Due to performance issues, subtree reduction (see subsection II-B) could not be applied in such a large dataset. Thus, simple tree reduction is used in its place. The mining step is performed through the ParSeMiS [15] implementation of CloseGraph, while InfoGain (ranking step) was implemented using WEKA [16].

The two other algorithms (ZhangShasha and pq-Grams) were implemented similarly, inserting a dataset reduction step before the graph mining step. Both implementations use the call trace selector of Figure 2, while different values of the n parameter are tested. The first implementation realizes the ZhangShasha algorithm and the second implementation the pq-Grams algorithm in order to reduce the size of the dataset.

All experiments were performed using an 8-core processor with 8 GB of memory. The graph reduction, dataset reduction and subgraph mining steps were performed in parallel. Graph reduction was performed on 8 threads, where each thread performed simple tree reduction to a fragment of the dataset. The TED algorithms were applied in parallel using 4 threads (using more threads was impossible due to memory limitations) that calculated the TED for each correct-incorrect pair of the dataset. Finally, CloseGraph was executed using 8 threads, while the trace parsing and ranking steps were sequential.

B. Experimental Results

The algorithms are evaluated both in terms of effectiveness and performance. Concerning certain parameters, p and q of the pq-Grams approach were given the values 2 and 3 respectively, having little impact on performance and effectiveness, and CloseGraph was run with a 10% support threshold.

The performance results are shown in Table III, where the NoTED approach is the one not using any TED algorithm to reduce the size of the dataset. Due to space limitations in paper length, the average measurements are shown for all three bugs, instead of separate ones for each bug. In terms of performance, both proposed implementations (pq-Grams and ZhangShasha) clearly outperform the NoTED approach. In particular, even when n equals 35, the pq-Grams algorithm requires no more than 10 minutes, whereas the NoTED approach requires almost 90 minutes. The ZhangShasha algorithm is also quite compelling requiring less than 15 minutes to run. Thus, the pq-Grams and ZhangShasha approaches are approximately 9.5 and 6.5 times faster than the NoTED approach, respectively.

Concerning all approaches, the mining step is indeed the most inefficient. Although ranking might also seem inefficient, its elapsed time depends mainly on the output of the mining step. Concerning the graph reduction step, simple tree reduction performs quite efficiently. Although graph reduction techniques deviate from the scope of this paper, note that subtree reduction required many hours to reduce the graphs.

Performance results are also shown in Figure 5b, where the vertical axis is in logarithmic scale in order to sufficiently illustrate the steps of the algorithms. As expected, performance is largely affected by the number of graphs taken into account, i.e., the n parameter. The impact of n is depicted in Figure 5a; the execution time of both approaches is high-order-polynomial with respect to consecutive values of n. This is expected since subgraph mining algorithms, such as CloseGraph, are affected by the size of the graphs and the size of the dataset. Further analyzing Figure 5a, the peak at $n = 25$ is not totally unexpected since the performance of subgraph mining algorithms may be affected by numerous properties, such as the structure of the graph. In any case, concerning the proposed algorithms, pq-Grams executes faster than ZhangShasha for all values of n, while NoTED is certainly less efficient.

Table IV provides effectiveness measurements for locating the three bugs, for all different algorithms. The “Position” attribute of the table indicates how many functions should the developer examine in order to locate the bug. This metric is created using the final ranking of the functions and identifying the position of the “buggy” function. Using the total number of functions, which for bugs 1, 2, and 3 is 637, 737, and 777 respectively, the percentage of the program’s functions that should be examined to locate the bug is also provided.

Our approaches seem to perform not only closely, but also even more effectively than the NoTED approach, as long as

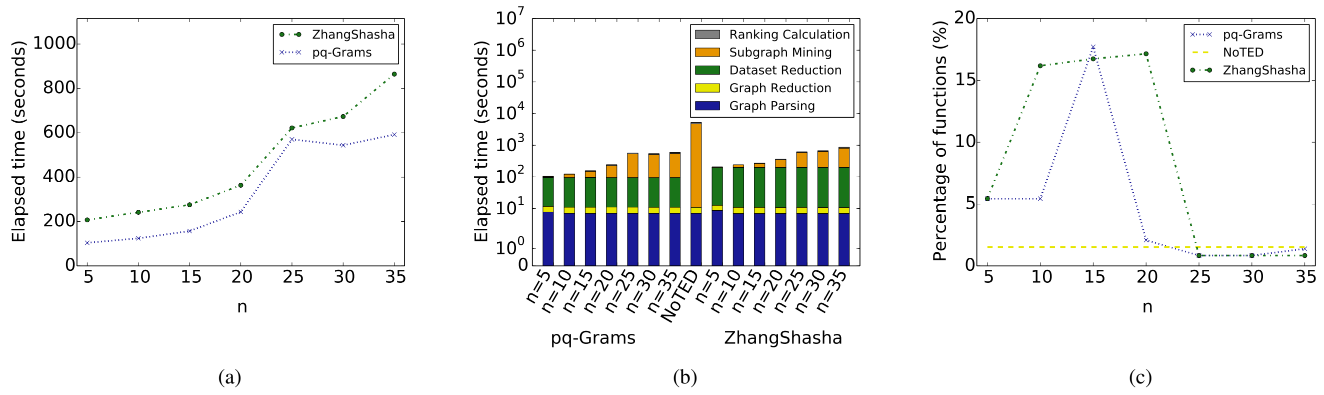


Fig. 5. Average performance and effectiveness diagrams for the bugs of the dataset. Diagrams (a) and (b) provide the elapsed time for each run of the algorithm. Diagram (a) depicts the total elapsed time of the pq-Grams and ZhangShasha approaches versus the value of parameter n (which denotes the number of traces retained from each of the two sets, correct and incorrect), while diagram (b) illustrates the performance for each phase of the algorithms in logarithmic scale. Diagram (c) illusrates the percentage of functions to be examined in order to detect the bug, versus n.

n is large enough. In fact, the pq-Grams and ZhangShasha approaches provide a better ranking for the third bug if n is greater than or equal to 25. Effectiveness is also satisfactory for the first two bugs. The diversity of the results for the three bugs is rather expected since the size of the traces is different for each bug (see Table II). Thus, the third bug produces a much more difficult test case than the other two.

The impact of n on effectiveness is illustrated in Figure 5c, which depicts the percentage of functions required to be examined versus n for the three implementations. The effectiveness of our algorithms is indeed significant for large enough values of n. Although small n values result in less satisfactory results, this is rather expected since useful trace information is lost. However, selecting an appropriate n value not only reaches but also surpasses the effectiveness of the NoTED algorithm.

VI. Conclusion and Future Work

Although there are several approaches for locating non-crashing bugs in source code, many of them suffer from scalability issues. With support from the experimental results of subsection V-B, we argue that our approaches achieve scalability without compromising effectiveness. According to our findings, reducing also the size of the dataset, as opposed to reducing only the graphs, yields quite promising results.

Concerning the dataset reduction step, both TED algorithms are very efficient. Although the performance of ZhangShasha is satisfactory, using pq-Grams provided faster runs and better function rankings. Conclusively, when only the relative edit distance of tree pairs is important, approximate TED algorithms, such as pq-Grams, perform similarly to exact ones.

The field of dynamic bug detection is far from exhausted when it concerns creating a scalable and effective algorithm. We argue, however, that our algorithms are a step in the right direction. Future research includes further testing to explore their efficiency in different datasets. In addition, further analysis of TED algorithms could lead to more effective solutions. Finally, the dataset reduction and subgraph mining steps can also be improved by designing new approaches. In any case, dataset reduction should definitely be taken into account.

References

[1] B. Liblit, A. Aiken, A. X. Zheng, and M. I. Jordan, “Bug isolation via remote program sampling,” SIGPLAN Not., vol. 38, no. 5, May 2003, pp. 141–154.

[2] C. Liu, X. Yan, L. Fei, J. Han, and S. P. Midkiff, “Sober: statistical model-based bug localization,” SIGSOFT Softw. Eng. Notes, vol. 30, no. 5, Sept. 2005, pp. 286–295.

[3] M. Renieris and S. Reiss, “Fault localization with nearest neighbor queries,” in Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on, 2003, pp. 30–39.

[4] C. Liu, X. Yan, H. Yu, J. Han, and P. S. Yu, “Mining Behavior Graphs for ”Backtrace” of Noncrashing Bugs,” in SDM, 2005.

[5] G. Di Fatta, S. Leue, and E. Stegantova, “Discriminative pattern mining in software fault detection,” in Proc. of the 3rd international workshop on Software quality assurance (SOQUA), 2006, pp. 62–69.

[6] F. Eichinger, K. Böhm, and M. Huber, “Mining edge-weighted call graphs to localise software bugs,” in European Conference on Machine Learning and Knowledge Discovery in Databases, 2008, pp. 333–348.

[7] X. Yan and J. Han, “gspan: Graph-based substructure pattern mining,” in Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM), 2002, pp. 721–724.

[8] Y. Chi, Y. Yang, and R. R. Muntz, “Indexing and mining free trees,” in Proc. of the Third IEEE International Conference on Data Mining, ser. ICDM ’03, 2003, pp. 509–512.

[9] X. Yan and J. Han, “Closegraph: mining closed frequent graph patterns,” in Proc. of the 9th ACM international conference on Knowledge Discovery and Data Mining, ser. KDD ’03, 2003, pp. 286–295.

[10] K.-C. Tai, “The tree-to-tree correction problem,” J. ACM, vol. 26, no. 3, July 1979, pp. 422–433.

[11] K. Zhang and D. Shasha, “Simple fast algorithms for the editing distance between trees and related problems,” SIAM J. Comput., vol. 18, no. 6, Dec. 1989, pp. 1245–1262.

[12] N. Augsten, M. Böhlen, and J. Gamper, “Approximate matching of hierarchical data using pq-grams,” in Proceedings of the 31st international conference on Very large data bases (VLDB), 2005, pp. 301–312.

[13] “daisydiff: A java library to compare html files,” [retrieved August, 2013]. [Online]. Available: http://code.google.com/p/daisydiff/

[14] “Software & algorithms, ISSEL,” [retrieved August, 2013]. [Online]. Available: http://issel.ee.auth.gr/software-algorithms/

[15] M. Philippsen, M. Wörlein, A. Dreweke, and T. Werth, “Parsemis: The parallel and sequential mining suite,” [retrieved August, 2013]. [Online]. Available: www2.informatik.uni-erlangen.de/EN/research/ParSeMiS/

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The weka data mining software: an update,” SIGKDD Explor. Newsl., vol. 11, no. 1, Nov. 2009, pp. 10–18.

Finding Common Subsequences in Recorded Test Cases

Martin Filipsky, Miroslav Bures and Ivan Jelinek
Department of Computer Science and Engineering
Czech Technical University in Prague
Prague, Czech Republic
{filipma2, buresm3, jelinek}@fel.cvut.cz

***Abstract*—Current trends in the agile software development prefer to deliver finished stories with automated tests, which results in a fact that many Quality assurance engineers struggle with the lack of time. Rapidly changing applications prevent them from finishing the automation by the end of the sprint as they cannot develop the tests in advance, and have to wait until the stable deliverable is done. Test recording might help them to resolve the problem as it offers very fast test automation in comparison to other approaches. However, it results in a very expensive and a time demanding test maintenance. In this paper, we present an approach that helps the engineers with the maintenance by introducing a concept of automatically detected reusable parts within the test recordings. Those reusable parts increase the efficiency of the test recording approach, remove its main drawbacks, and help to bring test recording closer to scripting approaches.**

Keywords-functional testing, test automation, test recording, genetic algorithm

I. Introduction

Test automation includes a couple of challenges [9]. Since testing teams are usually limited by finances, time as well as resources [3], they have to use simple but efficient approaches for the test harnessing. Here comes the test recording [5] in place as it allows creating automated tests quickly. On the other hand, this method is not generally understood as efficient due to its significant maintenance overhead [1].

In our recent research [6], we have proposed a framework for the test automation based on the test recording. We introduced a concept of reusable parts allowing simplifying the test maintenance. Introducing the reusable parts means to find common parts within the recorded tests. The problem of finding them can be transformed into the finding longest common subsequence problem [2].

The paper is organized as follows. Section 2 introduces the problem. Section 3 summarizes the previous results. In Section 4, we describe our solution of the problem. In Section 5, we conclude with outlines for future work.

II. The Problem

The Longest Common Subsequence (LCS) problem is defined as finding LCS common to all sequences in a set of sequences. The subsequence is a sequence that can be derived from another sequence by deleting some elements of the original sequence without changing the order of the remaining elements. Unlike the subsequences, the substrings cannot be derived from another string by deleting some elements.

Consider a string *S* = "AACECAACE", then following strings: (i) *S*, (ii) "AACAE", (iii) "CCCE", (iv) "ACECA" and (v) ε are subsequences of the string *S*. The subsequence "ACECA" is also the substring of the string *S* but the subsequences "AACAE" and "CCCE" are not. Now consider strings S_1 = ”**A**EB**EEB**C**C**B**A**C**A**” and S_2 = ”CE**A**C**E**B**EBCBAA**”. Then “**AEEBCBAA**” is the LCS of the given strings, which currently preserves the order of elements and allows deleting elements from the original strings.

The standard LCS problem is defined for finding a single LCS. However, if we need to find all subsequences with at least length *l*, the problem is getting more complex. In general, the decision, if a subsequence *w*, which is common to all sequences and has the length at least *l*, exists over an alphabet *Σ*, is an NP-complete problem [13]. To overcome this limitation, we are planning to employ an evolutionary computational technique to find LCS.

Understanding tests as sequences of steps might be more beneficial than understanding them as strings. Finding common subsequences (CS) might result in longer subsequences than finding common substrings. However, it brings the need to define conditions when a subsequence is valid when excluding some steps from the test case. Otherwise, it might happen that the found CS could not be executed independently as the some steps might depend on excluded steps. Therefore, the state of the application would not be identical for all steps within the common part.

When finding CS for informational purposes, all steps can be excluded. However, if we want to understand CS as functions (as we want to get closer to the scripting approach), we have to exclude all steps changing the state of the application (Fig. 1), i.e., only the passive (validation) steps can be interposed between the common sequence.

III. Related Work

Searching in structured data like test steps or test scripts represent challenges in the current computing. As the machine processing becomes more widely used in order to replace the human labor, standard approaches [10, 11] for the string searching introduced in 70's cannot be often easily employed for those data.

Unlike unstructured data, the structured data are organized in elements. However, the elements (tags) are not supposed to convey information, e.g., in Extensible Markup Language (XML).

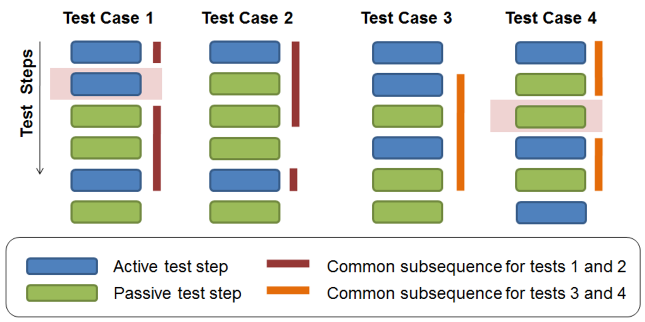


Figure 1. Two types of inserted steps (in light red)

Tags define a structure of the document. We talk about hybrid data when both types of data are in one document.

Zhu et al. [17] noticed that the text search on hybrid data may result in a bad ranking of the searching results. They demonstrated why the text search fails or gives insufficient results when used without considering the structured data.

XML can be seen as a good format for a test case representation. However, searching within those structured data requires special approaches, which can be divided into two categories: (i) information retrieval, and (ii) database-oriented. The database-oriented approach [12] is based on a decomposition of XML documents and their storage in relational databases. The drawback is a query processing, which may become expensive due to an excessive number of joins required to recover information from the fragmented data. The information retrieval approaches employ other computational techniques like genetic algorithms in several ways [16].

Srinivasa et al. [15] introduced an approach for an XML information retrieval mechanism. Based on keyword queries, they explored how to retrieve and rank XML fragments using Genetic Algorithms.

An evolutionary technique for the LCS problem is discussed in [7]. The genetic algorithm (GA) encodes candidate sequences as binary strings as long as the shortest of given string. Authors initialize conventionally random genotypes. They demonstrated that the algorithm always found an optimum solution, runs in reasonable times even on large instances, and achieves better results when compared to approaches based on the dynamic programming.

Julstrom and Hinkemeyer [8] noticed that GA might find good solutions more quickly in situations, when a problem is one of constrained optimization, and genotypes of the initial population are represented by empty solutions.

Finding longest common subsequences in strings is commonly solved by GAs or dynamic programming. The recent research shows that GAs achieve the best results in comparison to other approaches. Several research teams presented approaches finding the LCS in strings. Nevertheless, those approaches do not deal with structured and parameterized data represented by tests in different input alphabets. Since the current research in testing is mostly focused on the generation of test cases based on a code analysis [4], or on an analysis of regression test selection [14], we see a potential in the research of techniques for the maintenance of recorded tests to decrease costs for the test maintenance.

IV. Proposed Solution

In this section, we present individual parts of our approach. We start with mapping tests to strings. Then we present control parameters, outputs, and introduce our proposal of LCS solver. Finally, we explain step signatures.

A. *Mapping of Tests to Strings*

Current solutions for the LCS problem are proposed for strings (unstructured data). Since test cases are represented, e.g., in a domain-specific language (DSL), we need to adapt the current solutions to work with the structured data. Strings consist of single elements, i.e., characters, which form sequences. We plan to represent the test cases internally in the DSL (see Listing 1) describing tests, modules, objects, actions, etc. The Listing 1 shows the recorded user activity forming the sequence in the XML.

If we consider all child tags of the XML tag Step including their parameters and values, we will deal with high number of variables. It will result in a difficult mapping of the XML tag Step to a single character required by LCS solvers. On the other hand, if we consider just Step as one character, we can understand the tag as one character of the string. Therefore, the string will consist of complex units (Fig. 2). Such a representation enables working with structured data using conventional LCS solvers. However, this approach would be too simplified as the steps might be understood as identical. They do not have enough properties for the identification, since the tag id or tag name is not enough. Therefore, all steps mapped to the same character could not be recognized. To identify test steps, we introduce step signatures, which are supposed to replace a step description. Otherwise, we would have to choose between the full text search not recognizing two identical but parameterized steps, and the mapping of steps to characters not allowing distinguishing them.

Unlike test cases in DSL, the use of, for example, Java brings new challenges. First of all, steps represented by commands or functions of the scripting language have to be simplified. Consider the complexity of the comparison when counting with language-specific features, parameters etc. However, the simplified elements still should have signatures to describe them, which results in a need to find either direct mapping of commands to steps including signature definitions for every proposed language. Another option is to find a general mapping of a limited subset of commands to the intermediate layer (DSL), and propose one signature based on the DSL.

In our research, we plan to do more investigations in order to decide if it is better to work with the source code directly, or if it is worth to transform the source code to the DSL and then, to process this representation.

B. *Inputs and Outputs*

The LCS solver expects two kinds of input data: (i) raw input data intended for processing (test recordings), and (ii) control data driving the processing. For the finding LCS, we expect to provide the LCS solver, i.e., the GA, with the input files either in the DSL, or in direct source codes.

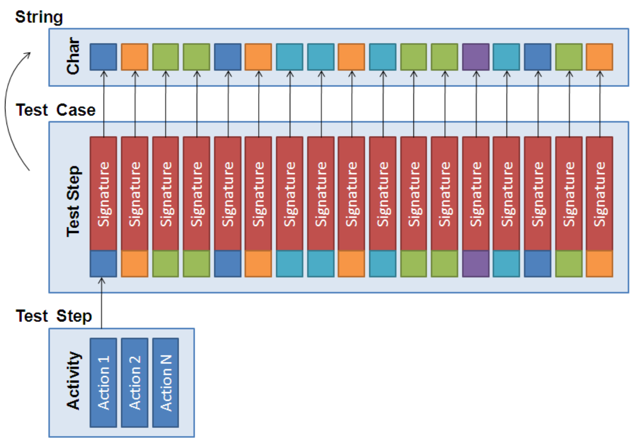


Figure 2. Mapping tests to simple strings

The condition is that the relevant mapping exists from test scripts to elements with signature. The output from the LCS solver should be a processed package of test recordings with identified common subsequences.

Since the LCS solver should be proposed to find the longest common subsequence as well as shorter CS in order to detect reusable parts, we need to provide the LCS solver with a threshold defining what lengths of common subsequences we are interested in. Moreover, we want the LCS solver to work with simple test step signatures and/or with the complex ones allowing recognizing identical but parameterized steps. In other words, the LCS is supposed to work at different level of details.

C. *Evolutionary Computations*

We based our solution on the approach presented in [8] and tailored the GA used in the LCS solver to fit our needs. For the LCS search, candidate sequences are encoded as binary strings as long as the shortest mapped given tests or the first given test if they are of the same length. If the element is present in the candidate sequence (in the chromosome), it is encoded by "1". If not, it is encoded by "0". Since [8] demonstrated that the GA achieves better results when the population is empty, i.e., the population is represented by zeros, we have decided not to employ any technique for the generation of the population.

Consider the example of three mapped tests T_1 = “**A** E B **E E B** C **C B**”, T_2 = “C E **A E E** C **B C B**”, and T_3 = “**A E E** E A **B C B** E“, and the chromosome c[*] = 1 0 0 1 1 1 0 1 1, then it means that T_1: c[i] = 1 is in the subsequence T_1[i], and T_1: c[i] = 0 is not in the subsequence T_1[i]. T_1 represents the shortest given test or the first test from tests with identical lengths.

Once GA finds a solution of the LCS problem, the LCS will be encoded in the chromosome. However, the found solution represents the LCS in one test, but does not define where to find the subsequence in other tests. We only know the mapping from the chromosome to T_1. Since the LCS solver is required to build a structure enabling to identify and access the subsequence in all tests, the computation of the LCS has to be followed up by another stage of computations finding the mapping.

The fitness function *v* is proposed to remunerate (1) long sequences, (2) the genotype whose subsequence is long as T_1, (3) strongly remunerate the genotype, in which the subsequence appears for each given test (4) strongly penalize the genotype whose subsequence is not found in any test. The fitness function cannot be positive unless the sequence appears in all given tests. Based on the assumptions above and the research of [8], the initial general fitness function is defined for every case as follows:

$$v = l + \alpha * m \quad (1)$$

$$v = v + \beta \quad (2)$$

$$v = \gamma * v \quad (3)$$

$$v = \delta * v * (K - m) \quad (4)$$

where *l* is a length of the subsequence, which *c[*]* represents, *m* is number of tests, which match with the subsequence, and *K* is the number of tests in the instance. The constants α, β, γ, and δ represent the parameters of the genetic algorithm and will be experimentally determined.

We are planning to employ several techniques for driving the evolution of the population, which will be divided into elite genotypes and the majority population. If the elite population does not change for several generations, some of the elite genotypes will be replaced by random genotypes to avoid local optimums. Remaining genotypes will be evolved using either a selective breeding of a position, or a mutation of the position. The genotypes to be modified will be selected by the tournament selection with the probability 1/*l*. We are intending to carry out additional investigations to decide which strategy would bring the best results.

D. *Signatures*

Steps of parameterized tests can be compared only in text mode. Therefore, we proposed signatures to help the steps get compared, and find common parts. Since the structure of the command might be variable (for example, consider commands with one, two, or more parameters), the usage of regular expressions would require to define regular expressions for all possible combinations to compare strings. Otherwise, the standard LCS solver could not compare parameterized data. Unlike the regular expressions, the signatures allow to define simplified ones for rough searches, and also detailed signatures for fine-grained searches. Moreover, they make use of the opportunity of the clear structure of the DSL (see Figure 3 representing a sample recorded test), and can be built in a simplified way for all commands than regular expressions.

```
<test case id=1 name="AddBob">
  <step id="1">
    <object id="Menu" type="Tree"\>
    <action name="Select" onFailure="">Tools;Login</action>
  </step>
  <step id="2">
    <object id="Username" type="InputBox"
      environment="Flex" \>
    <action name="Set" onFailure="">Alice</action>
  </step>
  <step id="4">
    <object id="Login" type="Button" environment="Flex" \>
    <action name="Click" onFailure="" \>
  </step>
</test case>
```

Figure 3. Recorded test case in the DSL

Let us explain the signatures on the example of the recorded test case captured in the DSL. The test case represents the login to the system. The base signature consists of descriptions of two entities (objects and actions). We proposed several levels of signatures for different needs. The Level 0 signatures are intended to represent subsequences of similar objects and actions. It provides the users with a possibility to find groups of similar commands independently of concrete objects. Level 1 is proposed for the standard LCS search. It enables to work with parameterized tests, but it is not so strict like Level 2, which finds absolute conformities of the subsequences including input values. Level 2 gives the user a choice, what attributes and parameters should be in the signature.

TABLE I. SIGNATURES

Level	Step	Signature
0	2	obj:inputbox&act:set
1	2	obj:inputbox.username&act:set
2	2	obj:inputbox.username+environment=flex &act:set+val:(hash)

The Table 1 presents the signatures for each level based on the sample recording (Figure 3) for the step 2. To simplify the signature as much as possible (consider long input data), the input parameters are replaced by hashes. The syntax of the signature is defined as follows:

obj:<type>{.<object_name>}{+<attribute>=<value>}&
act:<action_name>{+<parameter>:<value>}

where *obj* stands for the object entity, *act* represents the action entity, the *&* char links different entities. If more attributes are required to describe entities in the signature, they can be associated with the entity using the char "+". The entity attributes are not mandatory.

V. CONCLUSIONS AND FUTURE WORK

We have proposed the approach for finding LCS of test steps based on the evolutionary computational approach presented in [8]. Moreover, we proposed the method of the adaptation of the GA processing strings to process structured data represented by test cases. Furthermore, we introduced signatures for descriptions of steps, which currently enable finding LCS in different equivalence classes.

Our next goal of the research is to conduct experimental verifications of the proposed approach as well as to tune up the parameters of the GA. We are planning to compare results gained using the signatures to results gained using the regular expressions, and to find out the impact of different sizes of the input alphabet. One of our goals is also to confirm or disprove whether it is better transform inputs into the DSL, or if it is worth to work with test recordings directly without preprocessing.

Finally, we are planning to evaluate the results from several points of view. Firstly, we will check whether the results make sense, and whether found LCS would be similar to reusable units designed by human testers. Secondly, we will investigate the contribution of such approach with an emphasis on the efficiency of test automation and test maintenance.

REFERENCES

[1] B. R. Anand, H. Krishnankutty, K. Ramakrishnan, and V.C. Venkatesh, Business Rules-Based Test Automation: A Novel Approach for Accelerated Testing, White paper, Infosys, SETLabs Briefing, Special Issue April 2007, pp. 21-28.

[2] M. F. Balcan, CS 3510 – Design and Analysis of Algorithms, Lecture notes, Georgia College of Tech Computing, 2011.

[3] R. Black, Investing in Software Testing: The Cost of Software Quality, White paper, RBCS, 2000.

[4] Ugo Buy, Alessandro Orso, and Mauro Pezze. 2000. Automated Testing of Classes. In Proceedings of the 2000 ACM SIGSOFT international symposium on Software testing and analysis (ISSTA '00), ACM, New York, USA, pp. 39-48.

[5] M. Fewster and D. Graham, Software Test Automation: Effective Use of Test Execution Tools, Addison-Wesley Professional, ACM Press Books, September, 1999.

[6] M. Filipsky, M. Bures, and I. Jelinek, Framework for Better Efficiency of Automated Testing, In Proceedings The Seventh International Conference on Software Engineering Advances, Lisbon, Portugal, 2012, pp. 615-618.

[7] B. Hinkemeyer and B. A. Julstrom, A Genetic Algorithm for the Longest Common Subsequence Problem, In Proceedings of the 8th annual conference on Genetic and evolutionary computation, GECCO '06, ACM, New York, USA, 2006, pp. 609-610.

[8] B. Julstrom and B. Hinkemeyer. Starting From Scratch: Growing Longest Common Subsequences With Evolution. In Parallel Problem Solving from Nature - PPSN IX, vol. 4193 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 930-938.

[9] C. Kaner, Software Test Automation: A Real-World Problem, White paper, Los Altos Workshop on Software Testing 1-3, 1997-98.

[10] R. M. Karp and M. O. Rabin, Efficient Randomized Pattern-Matching Algorithms, IBM J. Res. Dev., vol. 31(2), March, 1987, pp. 249-260.

[11] D. E. Knuth, J. J. H. Morris, and V. R. Pratt, Fast Pattern Matching in Strings. SIAM Journal on Computing, vol. 6(2), 1977, pp. 323-350.

[12] R. W. Luk, H. V. Leong, T. S. Dillon, A. T. Chan, W. B. Croft, and J. Allan, A Survey in Indexing and Searching XML Documents, J. Am. Soc. Inf. Sci. Technol., vol. 53(6), May, 2002, pp. 415-437.

[13] D. Maier, The Complexity of Some Problems on Subsequences and Supersequences, Journal of the ACM 25, 1978, pp. 322-336.

[14] G. Rothermel and M. J. Harrold, “Analyzing Regression Test Selection Techniques,” IEEE Transactions on Software Engineering, vol. 22, pp. 529–551, August 1996.

[15] K. G. Srinivasa, S. Sharath, K. R. Venugopal, and L. M. Patnaik, Gaxsearch: An XML Information Retrieval Mechanism Using Genetic Algorithms, In Australian Conference on Artificial Intelligence, 2005, pp. 435-444.

[16] J. Yang and R. R. Korfhage, Effects of Query Term Weights Modification in Annual Document Retrieval: A Study Based on a Genetic Algorithm, In Proceedings of the Second Symposium on Document Analysis and Information Retrieval, 1993, pp. 271-285.

[17] H. Zhu, X. Yang, B. Wang, and Y. Wang, Improving Text Search on Hybrid Data, In Web-Age Information Management, vol. 7419 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 192-203.

Architecture-Based Conformance Testing

Elena Leroux, Flavio Oquendo, and Qin Xiong
IRISA, University of South-Brittany, France
E-mails: {elena.leroux | flavio.oquendo | qin.xiong}@irisa.fr

***Abstract*—In the last two decades, software architecture has played a central role in the development of software systems. It provides a high-level description for large-size and complex systems using suitable abstractions of the system's components and their interactions. In our work, the software architecture is described using a formal Architecture Description Language (ADL) designed in the ArchWare European Project, π-ADL-C&C. One of the purposes of this ADL is to allow formal validation of an implemented system with respect to its architectural model. In this paper, we propose a conformance testing approach for validating a software system with respect to its architecture. The architectural abstract test cases are derived from an Input-Output Symbolic Transition System (IOSTS) representing the architecture structure and behaviors, which are then translated into concrete test cases to be executed on the system under test. To illustrate our approach we use the coffee machine example.**

Keywords*—Software Architecture, Architecture Description Language, Architectural Conformance Testing, Validation

I. Introduction

During the past years a continuous growth, in size and complexity, of software and hardware systems has been observed. The problems, which were important in the pass, and which are related to a code development, e.g., the choice of data structure and algorithms, became less important than the ones related to the system design. This is not only due to the increased amount of code, but also to the need to distribute different components of the system and to have them interact in complex ways. To deal with these problems and to rise the level of abstraction at which software is conceived and developed, a software architecture has emerged. It was rapidly considered as an important sub-discipline of software engineering [1]. Software architecture allows developers: (1) to abstract away the details of the individual components of a system, (2) to represent a system as sets of components with associated connectors that describe the interactions (a) among these components, and (b) between the components and the environment, and (3) to guide the system design and evolution. In order to describe the software architecture of a system, a set of formal and semi-formal languages has been proposed [2], [3]. These ADLs help specify an architecture according to different viewpoints. The two following viewpoints are frequently used at a runtime perspective in the software architecture discipline.

The structural viewpoint is specified in terms of: (1) *components* (i.e., units of computation of a system), (2) *connectors* (interconnections among components for supporting their interactions), and (3) *configurations* of components and connectors. Thereby, an architecture description, from a structural viewpoint, should provide a formal specification of the architecture in terms of components and connectors, and how they are composed together.

The behavioral viewpoint is specified in terms of: (1) actions a system executes or participates in, (2) relations among actions to specify behaviors, and (3) behaviors of components and connectors, and how they interact.

An ADL challenge is the ability of a language to enable validation of designed systems very early in the software life cycle in addition to verification all along the software process. The π-ADL [4] language has been designed in order to meet this challenge. π-ADL is an executable specification language that allows formal description of software architectures of a system under development. A virtual machine of π-ADL runs specifications of the software architecture and enables its validation by simulation and testing as described in this paper.

The analysis and validation, by using, for example, software testing techniques, of software systems play a crucial role in the system development process. That is one of the reasons of the raising interest to the use of the architectural models in order to test systems behaviors with respect to their early architectural specification. *Software testing* [5] is a process consisting in the dynamic verification of system behaviors, which is performed by observing the execution of the system on a selected test case. Several contributions [6]–[13] have been proposed to tackle the problem of the validation of software systems by means of architectural testing. The brief overview of them is done in Section VI of this paper.

In this paper, we focus on *model-based conformance testing* [14], [15], which permits to derive test cases from a model representing the behavior of a software system, in order to check that this system fulfills its behavior. We use IOSTS as a model, which we generate from a formal architectural specification designed in the π-ADL language. The goal is to propose an approach for validation of software systems using their architectural specifications, and to illustrate its feasibility with a simple example.

The remainder of this paper is structured as follows: Section II presents the π-ADL language, which is used for architecture design, and a working example, used all along this paper, for the demonstration of our approach. Section III briefly describes the IOSTS formalism, which is used to model an architectural π-ADL specification and abstract test cases derived from this specification. Section IV presents our approach explaining how to generate test cases from a π-ADL architecture and execute them on a black-box system under test. Section V lists the tools used or/and developed to support our approach. Section VI summarizes our work, positions it with respect to the other works done in the field of the software architecture-based testing and gives a brief overview of related work. Section VII closes the paper with summary remarks.

II. The π-Architecture Description Language

In this section, we briefly present π-ADL, which we are using for the architecture description of a system under development, and we illustrate it with a working example of a coffee machine.

A. Overview

The π-ADL language [4], designed in the ArchWare European Project, is a formal, well-founded theoretically language based on the higher-order typed π-calculus [16]. It supports description of software architectures from a runtime perspective. Moreover, π-ADL has a virtual machine allowing execution of architectural specifications, and therefore, the validation of a software architecture by simulation is enabled. In the following, we briefly explain how the π-ADL language can be used for the formal definition of a software architecture.

In π-ADL, an architecture is described in terms of components, connectors, and their composition.

Components are described in terms of external ports and an internal behavior. Their architectural role is to specify computational elements of a software system. The focus is on computation to deliver system functionality. *Ports* are described in terms of connections between a component and its environment. Their architectural role is to put together connections providing an interface between the component and its environment. Protocols may be enforced by ports and among ports.

Connectors are basic interaction points. Their architectural role is to provide communication channels between two architectural elements. A component can send or receive values via connections. They can be declared as output connections (values can only be sent), input connections (values can only be received), or input-output connections (values can be sent or received).

From a black-box perspective, only ports (with their connections) of components and connectors and values passing through connections are observable. From a white-box perspective, internal behaviors are also observable.

π-ADL consists of a family of related ADLs. The π-ADL-C&C language describes an architecture at an abstract high level. This language is user-friendly, and it allows rapid design of architectures using the notions of component and connector. The π-ADL-Spec language is a canonical form of π-ADL. Finally, the π-ADL.NET language is a low level ADL, that makes possible an execution of architectural specification as it is equipped with a virtual machine.

B. Working Example

In this section, we present a working example of a simple coffee machine, which will be used all along the paper. Fig.1 shows the abstract architecture of the coffee machine in terms of components and connectors. This coffee machine accepts coins (thought the *Coin(Natural)* connector), the request for a beverage (thought the *PressButton()* connector), and the request for a command canceling (thought the *Cancel()* connector), and then either delivers the beverage (thought the *Deliver()* connector) or returns money back (thought the *Return(Natural)* connector). It consists of two components: *Payment* and *Beverage.*

A request for a beverage is received by the *Beverage* component from the user interface of the coffee machine. The purpose of this component is (1) to stock the information about the availability and the price of a coffee, (2) to wait until the beverage button is pressed, (2) to communicate the price to the *Payment* component, (3) to prepare a coffee, and (4) to deliver it to a customer. The *Beverage* component serves the coffee whenever the two following conditions are satisfied: first, a customer has paid enough (this information should be received from the *Payment* component), and second, coffee is not out

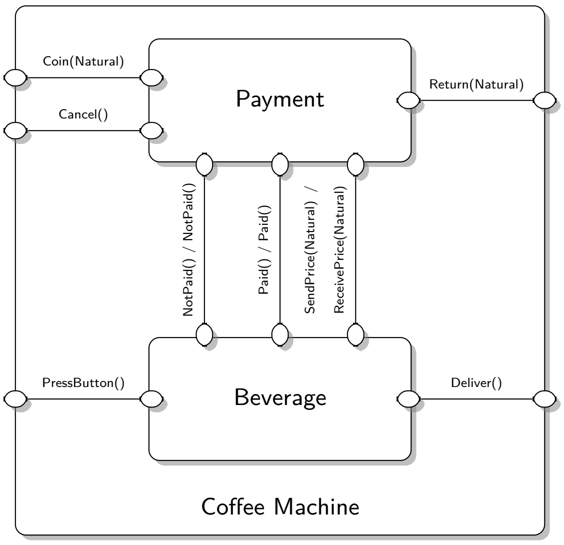


Fig. 1. The coffee machine architecture.

of stock. If the first condition is not satisfied, the component *Beverage* waits for another request for coffee and then checks again if the payment is sufficient. If the second condition is not satisfied, then the delivery of coffee is impossible, and the *Beverage* component is blocked.

The requests for a payment and for a command canceling coming from the user interface of the coffee machine are accepted by the *Payment* component. This component allows (1) to memorize the amount of money already paid by the customer, the number of coins inserted into the coffee machine, and the price of a coffee received form the *Beverage* component, (2) to communicate the information about sufficient/insufficient payment to the *Beverage* component, (3) to return the money back if the *Cancel* button has been pressed, or if the customer inserted more coins than authorized by the coffee machine, and (4) to return the difference between the price and the paid amount in the case of a coffee delivery.

Note that, the *Beverage* and *Payment* components communicate not only with their environment, but also with themself. Indeed, the *Beverage* component sends the price of a coffee through the *SendPrice(Natural)* connector to the *Payment* component. The latter receives the price through the *ReceivePrice(Natural)* connector. Moreover, the *Payment* component notifies the *Beverage* component if the customer has paid enough or not using the *Paid()* and *NotPaid()* connectors.

C. Architecture Description using π-ADL-C&C

In the previous section, we have informally described the structure and behavior of the coffee machine. In this section, we explain how this structure and behavior can be formalized using the π-ADL-C&C language. We begin with the description of two components of the coffee machine, namely the *Beverage* (see Fig.2) and the *Payment* (see Fig.3) components.

1) The beverage component. The *Beverage* component, shown on Fig.2, is declared as an abstraction (see line 1) with two *Natural* parameters: (1) *cBeverageQuantity* indicating the quantity of the beverage in the coffee machine, (2) *cPrice* indicating the price of the beverage. The external ports of this component are shown on lines 3-9, and described in terms

```
 1 component Beverage is abstraction(cBeverageQuantity : Natural, cPrice : Natural){
 2   port is {
 3     connection PressButton is in().
 4     connection Deliver is out().
 5     connection SendPrice is out(Natural).
 6     connection Paid is in().
 7     connection NotPaid is in().
 8   }
 9   drink is abstraction(vBeverageQuantity : location[Natural]){
10     if (vBeverageQuantity >= cBeverageQuantity) then{
11       via PressButton receive.
12       drink(vBeverageQuantity)
13     }else{
14       via PressButton receive.
15       via SendPrice send cPrice.
16       choose{
17         via NotPaid receive.
18         drink(vBeverageQuantity)
19       or
20         via Paid receive.
21         via Deliver send.
22         vBeverageQuantity := vBeverageQuantity'+1.
23         drink(vBeverageQuantity)
24       }
25     }
26   }.
27   behaviour is {
28     drink(location(0))
29   }
30 }
```

Fig. 2. The beverage component expressed in π-ADL-C&C.

of connections: *PressButton*, *Paid*, *NotPaid*, and *SendPrice*, *Deliver*, where the three first connections permit to receive the information from the environment (they are declared as input connections by using the keyword **in**) and the two last ones allow to send the information to the environment (they are declared as output connections by using the keyword **out**). Notice that, the *SendPrice* connection permits to send one value of the *Natural* type (see line 6) in order to be able to communicate the price of the beverage.

```
 1 component Payment is abstraction(cCoinNumber: Natural){
 2  port is {
 3     connection Coin is in (Natural).
 4     connection Return is out (Natural).
 5     connection Cancel is in ().
 6     connection ReceivePrice is in (Natural).
 7     connection Paid is out ()
 8     connection NotPaid is out ()
 9  }.
10  paying is abstraction(
11     cCoinNumber: Natural,
12     vPaid: location[Natural],
13     vCoinNumber: location[Natural],
14     vPrice: location[Natural]
15   ){
16    choose {
17      if vCoinNumber < cCoinNumber then {
18        via Coin receive pCoin : Natural.
19        vPaid := vPaid'+pCoin.
20        vCoinNumber := vCoinNumber'+1.
21        paying(cCoinNumber, vPaid, vCoinNumber, vPrice)
22      } else {
23        via Return send vPaid.
24        paying(cCoinNumber, location(0), location(0), location(0))
25      }
26    or
27      via ReceivePrice receive pPrice : Natural.
28      vPrice := pPrice.
29      paying(cCoinNumber, vPaid, vCoinNumber, vPrice)
30    or
31      via Cancel receive.
32      via Return send vPaid.
33      paying(cCoinNumber, location(0), location(0), location(0))
34    or
35      if vPaid >= vPrice then {
36        via Paid send.
37        via Return send (vPaid-vPrice).
38        paying(cCoinNumber, location(0), location(0), location(0))
39      } else {
40        via NotPaid send.
41        paying(cCoinNumber, vPaid, vCoinNumber, vPrice)
42      }
43    }
44  }.
45  behaviour is {
46    paying(cCoinNumber, location(0), location(0), location(0))
47  }
48 }
```

Fig. 3. The payment component expressed in π-ADL-C&C.

The behavior of the *Beverage* component is shown on lines 27-29, and described as a call to the *drink* abstraction carrying 0. The value 0 initializes the variable *vBeverageQuantity* memorizing the quantity of beverage already used. The body of the *drink* abstraction describes formally the behavior of the *Beverage* component of the coffee machine, explained informally in Section II-B. More precisely, the *Beverage* component verifies if the quantity of beverage is sufficient or not (see line 10). In the both cases above, it lets the customer to press the button (see lines 11 and 14), but (1) in the last case (the quantity of beverage is insufficient), the component is blocked (see the call to the same abstraction *drink* with the same value of parameter *vBeverageQuantity* on line 12), while (2) in the first case (the quantity of beverage is sufficient), the component communicates the price of the beverage using the *SendPrice* connection (see line 15), and then: (a) either returns into its initial state (see the call to the abstraction *drink* on line 18), if it has received the notification of insufficient payment through the *NotPaid* connection (see line 17), or (b) delivers the beverage using the *Deliver* connection (see line 21) and increases *vBeverageQuantity* by one (see line 22), if it has received the notification of sufficient payment through the *Paid* connection (see line 20), and comes back to its initial state (see the call to the abstraction *drink* on line 23).

2) The payment component. The formal description of the *Payment* component is given on Fig.3 and is similar to one of the *Beverage* component. Therefore, we do not detail it.

3) The architecture of the coffee machine. The architecture of the coffee machine is formally described in Fig.4. It is an abstraction whose behavior (see 2-12) is composed of two instantiated components *Beverage(10,3)* and *Payment(10)* (see lines 3-7). These components communicate via the unified connections shown on lines 8-10.

```
 1 architecture CoffeeMachine is abstraction() {
 2  behaviour is {
 3    compose{
 4      beverage is Beverage(10, 3)
 5    and
 6      payment is Payment(10)
 7    } where {
 8      payment::ReceivePrice unifies beverage::SendPrice and
 9      payment::Paid unifies beverage::Paid and
10      payment::NotPaid unifies beverage::NotPaid
11    }
12  }
13 }
```

Fig. 4. The architecture of a coffee machine in π-ADL-C&C.

III. Underlying Model for Test Case Generation

In this paper, we are interested in conformance testing of a system under development with respect to its architectural specification expressed at the user-level using π-ADL-C&C language. For test cases generation using STG [17], [18], we automatically translate a high-level architectural specification into the low-level model called IOSTS. We use IOSTS for describing architectural specifications, test purposes, and test cases, and assume that the black-box implementation can be described by an IOSTS of which only the external interface is known. The formal syntax and semantics of IOSTS are defined in [19]. The intuitive explanation is given below using the example depicted in Fig.5, which represents the payment component of the coffee machine. Notice that, the beverage component can also be modeled by IOSTS as it is shown in Fig.5.

An IOSTS is made up of *locations*, for example, p_1, p_2, p_3 and p_4, where p_1 is the *initial location*, and *transitions*.

The transitions are labeled with *actions*, *guards*, and *variable assignments*. For example, the transition with origin p_2 and destination p_2 has the guard ($vCoinNumber < cCoinNumber$), the input action *Coin?* carrying the data *pCoin* from the environment, and two variable assignments $vPaid := vPaid{+}pCoin$ and $vCoinNumber++$. The set of actions is partitioned into three disjoint subsets of *input*, *output*, and *internal* actions. The input/output actions interact with the environment and may carry data from/to it, while internal actions are used for internal computations. By convention, the names of input (*resp.* output) actions end with "?" (*resp.* "!"). The IOSTS in Fig.5 has two inputs: *Coin?* and *Cancel?*, three outputs: *Paid!*, *NotPaid!*, *Return!*, and one internal action: $\tau_{init\_payment}$. It operates with symbolic data consisting of *variables*, *constants*, and *parameters*. Intuitively, *variables* are data to compute with, *constants* are symbolic constants, and *parameters* are data to communicate with the environment. Note that the scope of parameters is only a transition labeled by an action, which carries these parameters. Thus, if the value of a parameter should be used in later computations, it should be memorized through an assignment to a variable.

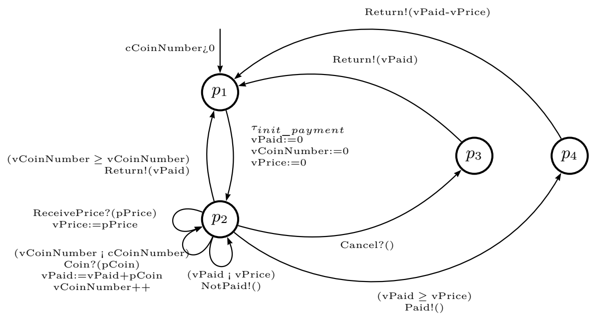


Fig. 5. The payment component modelled by an IOSTS.

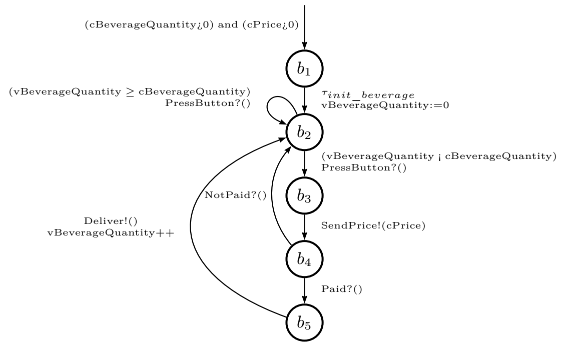


Fig. 6. The beverage component modelled by an IOSTS.

Informal semantics. Consider the IOSTS (*cf.* Fig.5) representing the *Payment* component of the coffee machine. The payment starts in the location p_1 with some value of the *cCoinNumber* constant satisfying the initial condition $cCoinNumber > 0$, that is, the number of coins accepted by the coffee machine is strictly positive. Then, it fires the transition labeled by the internal action $\tau_{init\_payment}$, assigns the three variables: *vPaid* storing the amount already paid, *vCoinNumber* memorizing the number of coins inserted into the machine, and *vPrice* storing the price of the beverage, to 0, and reaches the location p_2. Next, the *Payment* component expects either:

– a coin, denoted by the *Coin?* input action that carries in the *pCoin* parameter the value of the inserted coin. The variables *vPaid* and *vCoinNumber* are increased respectively by *pCoin* and by 1. Note that the *Coin?* action can be executed only in the case, where the number of the already inserted coins is less than the value of the *cCoinNumber* constant. Otherwise, the payment component returns the amount already paid (through the *Return!(vPaid)* output action) and moves back to the initial location p_1. Or
– the price of a beverage, denoted by the *ReceivePrice?* input action that carries in *pPrice* the cost of the beverage, the variable *vPrice* is initialized to the value of *pPrice*.

In the two cases above, the machine stays in the location p_2. If the payment is enough, i.e., $vPaid \geq pPrice$, the payment component, first of all, emits the *Paid!()* output action and moves to the location p_4, and then returns (through the *Return!(pPrice − vPaid)* output action) the difference between the paid amount and the cost of a beverage, i.e., *pPrice − vPaid*, and moves to the initial location p_1. Otherwise, the payment component sends the *NotPaid!()* output action and stays in the location p_2. Note that in the location p_2, the *Cancel?* input action can be received, which signifies that the *Cancel* button has been pressed. In this case, the payment component returns the amount already paid (through the *Return!(vPaid)* output action) and moves back to the initial location p_1.

Formal semantics. A *state* s is a pair $\langle l, \vartheta \rangle$, where l is a location and ϑ is a valuation of the constants and variables, e.g., $s = \langle Coin,\ cCoinNumber{=}10,\ vPrice{=}3,\ vPaid{=}2,\ vCoinNumber{=}4 \rangle$. An *initial state* $s^0 = \langle l^0, \vartheta^0 \rangle$ is a state where l^0 is the initial location, and ϑ^0 is a valuation of the constants and variables which satisfy the initial condition. We denote by S (*resp.* S^0) the set of all states (*resp.* initial states). A *valued action* α is a pair $\langle a, \omega \rangle$, where a is an action and ω is a valuation of the parameters of a, e.g., $\alpha = \langle Coin, pCoin = 1 \rangle$ or $\alpha = \langle \tau_{init\_payment} \rangle$. We denote by $\Lambda = \Lambda^{?} \cup \Lambda^{!} \cup \Lambda^{\tau}$ the set of valued actions, which is partitioned into three subsets of valued input, valued output, and internal actions. Next, we define the *transition relation* $\rightarrow$ as the set of triples $\langle s, \alpha, s' \rangle$, where $s = \langle l, \vartheta \rangle$, $s' = \langle l', \vartheta' \rangle$ are states and $\alpha = \langle a, \omega \rangle$ is a valued action. Here, (1) ϑ and ω are valuations of the constants, variables, and parameters, which satisfy the guard of a transition t with the origin l and the destination l' that is labeled with the action a, and (2) ϑ' is the new valuation of the variables and constants obtained from ϑ by the variable assignments of t.

Definition 1: A *behavior* β is a sequence of states and valued actions starting from an initial state and following the transition relation, i.e., $\beta : \quad s^0 \xrightarrow{\alpha_1} s_1 \xrightarrow{\alpha_2} s_2 \ldots s_{n-1} \xrightarrow{\alpha_n} s_n$ where $\rightarrow$ is the transition relation, $s^0 \in S^0$, and for all $i \in [1, n]$: $s_i \in S$, $\alpha_i \in \Lambda$.

To describe observable behaviors of IOSTS we define the relation $\Rightarrow$ as follows:

– $s \overset{\varepsilon}{\Rightarrow} s' \triangleq (s = s') \vee (\exists s_0, \ldots, s_n \in S.\ s = s_0 \xrightarrow{\tau_1} s_1 \ldots s_{n-1} \xrightarrow{\tau_n} s_n = s')$, where for all $i \in [1, n]$: $\tau_i \in \Lambda^{\tau}$;
– $s \overset{\alpha}{\Rightarrow} s' \triangleq \exists s_1, s_2 \in S.\ s \overset{\varepsilon}{\Rightarrow} s_1 \xrightarrow{\alpha} s_2 \overset{\varepsilon}{\Rightarrow} s'$, where $\alpha \in \Lambda^{?} \cup \Lambda^{!}$.

Definition 2: An *observable behavior* β is a sequence of states and valued input or output actions, i.e., $\beta : \quad s^0 \overset{\alpha_1}{\Rightarrow} s_1 \overset{\alpha_2}{\Rightarrow} s_2 \ldots s_{n-1} \overset{\alpha_n}{\Rightarrow} s_n$ where $s^0 \in S^0$, and for all $i \in [1, n]$: $s_i \in S,\ \alpha_i \in \Lambda^{?} \cup \Lambda^{!}$.

Definition 3: A *trace* σ is the sub-sequence of an observable behavior $\beta : s^0 \overset{\alpha_1}{\Rightarrow} s_1 \overset{\alpha_2}{\Rightarrow} s_2 \ldots s_{n-1} \overset{\alpha_n}{\Rightarrow} s_n$, which consists of

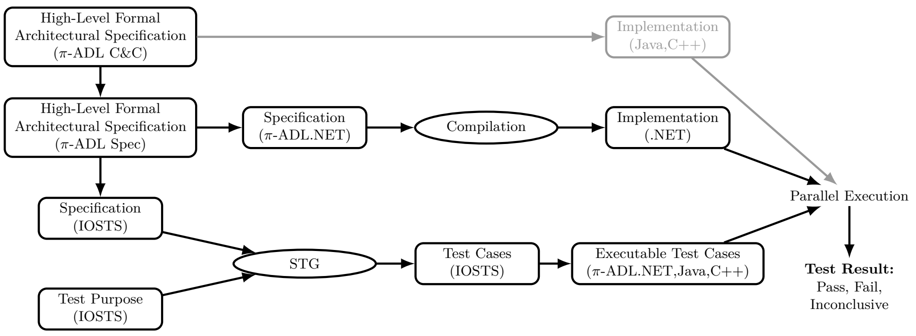

High-Level Formal Architectural Specification (π-ADL C&C)
Implementation (Java,C++)
High-Level Formal Architectural Specification (π-ADL Spec)
Specification (π-ADL.NET)
Compilation
Implementation (.NET)
Parallel Execution
Specification (IOSTS)
STG
Test Cases (IOSTS)
Executable Test Cases (π-ADL.NET,Java,C++)
Test Result: Pass, Fail, Inconclusive
Test Purpose (IOSTS)

Fig. 7. Outline of the approach.

valued input or output actions, i.e., $\sigma : \alpha_1\alpha_2\dots\alpha_n$ where for all $i \in [1, n]$: $\alpha_i \in \Lambda^? \cup \Lambda^!$.

A. Conformance Relation

The conformance relation defines the set of system's implementations which are correct with respect to its architectural specification. Intuitively, an implementation is conformant to a specification if for each trace of the specification, the implementation produces only outputs, which are allowed by the specification. To define the conformance relation formally, we first define the set of states in which an IOSTS M can be after the observable trace σ: $(M\ after\ \sigma) \triangleq \{s \in S \mid \exists s^0 \in S^0.\ s^0 \overset{\sigma}{\Rightarrow} s\}$, and the set of valued output (*resp.* input) actions which can be generated by M when it is in some state s among the set of states $\tilde{S}$: $Out(\tilde{S}) \triangleq \{\alpha \in \Lambda^! \mid \exists s \in \tilde{S}.\ s \xrightarrow{\alpha}\}$ (*resp.* $In(\tilde{S}) \triangleq \{\alpha \in \Lambda^? \mid \exists s \in \tilde{S}.\ s \xrightarrow{\alpha}\}$), where $s \xrightarrow{\alpha} \ \triangleq \exists s' \in S.\ s \xrightarrow{\alpha} s'$. Finally, denote by $Traces(M)$ the set of traces of M. Note that if a trace σ does not belong to $Traces(M)$ then $Out(M\ after\ \sigma)$ and $In(M\ after\ \sigma)$ are the empty set. For two IOSTS M_1, M_2 and each trace $\sigma \in Traces(M_1) \setminus Traces(M_2)$ we define $Out(M_2\ after\ \sigma)$ and $In(M_2\ after\ \sigma)$ to be the empty set.

Definition 4: The *conformance relation* between two IOSTS *IUT* and *Spec* with fixed, identical constants is defined as follows: $(IUT\ \mathbf{conf}\ Spec) \triangleq \forall \sigma \in Traces(Spec).Out(IUT\ after\ \sigma) \subseteq Out(Spec\ after\ \sigma)$.

IV. Approach for Architecture Validation

In this section, we describe the approach, which we use for the architecture validation of a system under development. This approach is depicted in the Fig.7 and presented below.

A. From π-ADL-C&C to π-ADL-Spec

The first step of our approach consists in the transformation of a high-level architectural specification described in π-ADL-C&C into its canonical form in π-ADL-Spec. To illustrate this transformation we use the payment component whose π-ADL-C&C code is shown in Fig.3. The result of the transformation is shown on Fig.8.

a) The components and their internal behaviors declared as abstractions are translated into the individual abstractions of behaviors. These individual abstractions can be later instantiated as behaviors by an application. Moreover, to enable a recursive call of an abstraction instance, this abstraction should be declared as a recursive abstraction in the π-ADL-Spec language by using the keyword "`recursive`". For example, the payment component (see lines 1-44 of Fig.3) corresponds to its individual abstraction shown on lines 43-45 of Fig.8; and its internal behavior "paying" (see lines 12-45 of Fig.3) corresponds to the recursive abstraction shown on lines 1-42 of Fig.8. Notice that, the parameters of components and internal behaviors are the same as the parameters of the corresponding individual abstractions. See, for example, the line 1 of Fig.3 and the corresponding line 43 of Fig.8.

```
 1 recursive value paying = abstraction(
 2   cCoinNumber: Natural,
 3   vPaid: location[Natural],
 4   vCoinNumber: location[Natural],
 5   vPrice: location[Natural]
 6 ){
 7   value Coin = connection(Natural);
 8   value Return = connection(Natural);
 9   value Cancel = connection();
10   value ReceivePrice = connection(Natural);
11   value Paid = connection();
12   value NotPaid = connection();
13
14   choose{
15     if('vCoinNumber < cCoinNumber) then{
16       via Coin receive pCoin : Natural;
17       vPaid := 'vPaid+pCoin;
18       vCoinNumber := 'vCoinNumber+1;
19       paying(cCoinNumber, vPaid, vCoinNumber, vPrice)
20     } else {
21       via Return send vPaid;
22       paying(cCoinNumber, location(0), location(0), location(0))
23     }
24     or
25       via ReceivePrice receive pPrice : Natural;
26       vPrice := pPrice;
27       paying(cCoinNumber, vPaid, vCoinNumber, vPrice)
28     or
29       via Cancel receive;
30       via Return send 'vPaid;
31       paying(cCoinNumber, location(0), location(0), location(0))
32     or
33       if('vPaid >= 'vPrice) then{
34         via Paid send;
35         via Return send ('vPaid-'vPrice);
36         paying(cCoinNumber, location(0), location(0), location(0))
37       } else {
38         via NotPaid send;
39         paying(cCoinNumber, vPaid, vCoinNumber, vPrice)
40       }
41   }
42 };
43 value Payment = abstraction(cCoinNumber: Natural){
44   paying(cCoinNumber, location(0), location(0), location(0))
45 }
```

Fig. 8. The payment component expressed in π-ADL-Spec.

b) The connections, declared in a π-ADL-C&C component (see for example, lines 3-8 of Fig.3), should be declared in the scope of a π-ADL-Spec abstraction in which they are used (see lines 7-12 of Fig.8). Notice that, the syntax for the declaration of a connection has been changed. Moreover, in the π-ADL-Spec language we do not need to specify if the connection is used to receive or to send information from/to its environment.

B. From π-ADL-Spec to π-ADL.NET

In order to obtain a system ready to be compiled and executed, we need to transform the π-ADL-Spec specification into the π-ADL.NET code. This section briefly outlines some important points of this transformation (see Fig. 8 and 9).

a) For each abstraction of π-ADL-Spec, its list of parameters, containing more than one parameter (see for example, lines 2-5 of Fig.8), is encapsulated as a value of the view type in the π-ADL.NET code (see respectively lines 1-5 of Fig.9). Each value of the view type $\texttt{view[label}_1\texttt{:T}_1\texttt{,...,label}_n\texttt{:T}_n\texttt{]}$ is a view $\texttt{view(label}_1\texttt{=v}_1\texttt{,...,label}_n\texttt{=v}_n\texttt{)}$, where for $i \in [1, n]$, each value v_i has type T_i, and each label $label_i$ has the same name as its corresponding parameter in the π-ADL-Spec code. The reason is that the π-ADL.NET language does not support a list of parameters for a value passing.

```
 1 value paying is abstraction(args:view[
 2   cCoinNumber: Integer,
 3   vPaid: Integer,
 4   vCoinNumber: Integer,
 5   vPrice: Integer]
 6 ){
 7   Coin : connection[Integer];
 8   Return : connection[Integer];
 9   Cancel : connection[Void];
10   ReceivePrice : connection[Integer];
11   Paid : connection[Void];
12   NotPaid : connection[Void];
13   pCoin : Integer;
14 
15   choose {
16     if (args::vCoinNumber < args::cCoinNumber) do {
17       via Coin receive pCoin;
18       args::vPaid = args::vPaid+pCoin;
19       args::vCoinNumber = args::vCoinNumber+1;
20       via paying send view(cCoinNumber:args::cCoinNumber, vPaid:args::vPaid,
              vCoinNumber:args::vCoinNumber, vPrice:args::vPrice);
21     } else do{
22       via Return send vPaid;
23       via paying send view(cCoinNumber:args::cCoinNumber, vPaid:0, vCoinNumber:0,
              vPrice:0);
24     }
25   or
26     via ReceivePrice receive pPrice : Natural;
27     vPrice = pPrice;
28     via paying send view(cCoinNumber:args::cCoinNumber, vPaid:args::vPaid,
            vCoinNumber:args::vCoinNumber, vPrice:args::vPrice);
29   or
30     via Cancel receive;
31     via Return send vPaid;
32     via paying send view(cCoinNumber:args::cCoinNumber, vPaid:0, vCoinNumber:0, vPrice:0);
33   or
34     if (vPaid >= vPrice) do {
35       via Paid send;
36       via Return send (vPaid-vPrice);
37       via paying send view(cCoinNumber:args::cCoinNumber, vPaid:0, vCoinNumber:0,
              vPrice:0);
38     } else do {
39       via NotPaid send;
40       via paying send view(cCoinNumber:args::cCoinNumber, vPaid:args::vPaid,
              vCoinNumber:args::vCoinNumber, vPrice:args::vPrice);
41     }
42   }
43 };
44 value Payment is abstraction(cCoinNumber: Integer){
45   via paying send view(cCoinNumber:args::cCoinNumber, vPaid:0, vCoinNumber:0, vPrice:0);
46 }
```

Fig. 9. The payment component expressed in π-ADL.NET.

b) Each call to a π-ADL-Spec abstraction carrying parameters, which permit to establish the communications between behaviors and abstractions (see for example, line 19 of Fig.8), is transformed, in the π-ADL.NET code, into an output action sending these parameters via the connection with the same name as the corresponding π-ADL-Spec abstraction (see line 20 of Fig.9).

c) Each location type in the π-ADL-Spec language (see for example, line 3 of Fig.8) is transformed into the type of the value stored in this location (see line 3 of Fig.9).

C. From Architectural Specification to Implementation

The goal of this step of our approach is to obtain an executable software system. To reach this goal we use the π-ADL compiler [20] developed in C# by Z.Qayyum, and executable on .NET platform. This compiler takes as input a π-ADL.NET code and transforms it into an executable system. We then run this system on a persistent virtual machine developed for executing architectural descriptions based on the operational semantics of π-ADL.

D. From π-ADL-Spec to IOSTS

In this section, we informally describe the transformation of an architectural specification expressed in π-ADL-Spec into its IOSTS model. We use the example of the payment component, shown in Fig.8 and called $S_{\pi\text{-ADL-Spec}}$, in order to illustrate this transformation, which results in the IOSTS, depicted in Fig.5 and called S_{IOSTS}.

a) Each π-ADL-Spec abstraction corresponds to one IOSTS model. For example, the abstraction shown on lines 44-46 of $S_{\pi\text{-ADL-Spec}}$ corresponds to S_{IOSTS} modeling behaviors of the payment component of the coffee machine.

b) The connections of a π-ADL-Spec abstraction become the input/output actions of the corresponding IOSTS. For example, the connections of $S_{\pi\text{-ADL-Spec}}$, i.e., *Coin*, *Cancel*, and *Return*, *Paid*, *NotPaid* (see lines 7-12), are the input/output actions of S_{IOSTS}.

c) Each input and output prefix, whose respective syntax is "**`via connection receive value`**" and "**`via connection send value`**", of a π-ADL-Spec abstraction is transformed into a transition of IOSTS labeled with an action corresponding to `connection` carrying out parameters corresponding to **`value`** of this prefix. Each silent prefix, indicated by the keyword "**`unobservable`**", is translated to a transition of IOSTS labeled with an internal action. Notice that, all the assignments following the prefix become assignments of the transition corresponding to this prefix. Moreover, if the prefix is surrounded with the "**`if(condition) then`**`{...}`" structure, then its corresponding, in the IOSTS model, transition is guarded by `condition` mentioned in this structure. For example, the π-ADL-Spec code of lines 15-23 corresponds to two transitions of S_{IOSTS} leaving from the location p_2 and labelled with the *Coin?* and *Return!* actions.

d) A sequence of input, output, and silent prefixes in the π-ADL-Spec language is modeled by the sequence of the corresponding transitions in the IOSTS model. For example, the sequence "**`via Cancel receive.via Return send 'vPaid`**" of $S_{\pi\text{-ADL-Spec}}$ (see lines 29-30) is represented by two consequent transitions $(p_2, Cancel?(), p_3)$. $(p_3, Return!(pPaid), p_1)$ of S_{IOSTS} (see Fig.5).

e) The "**`choice`**" structure of π-ADL-Spec permits to model a location of an IOSTS with several outgoing transitions. For example, the code of lines 14-41 of $S_{\pi\text{-ADL-Spec}}$ corresponds to p_2 of S_{IOSTS} and to six transitions outgoing from p_2.

f) A call to an abstraction in the π-ADL-Spec language, means that the transition corresponding to a prefix preceded by this call, should be redirected to one of already created locations of the IOSTS. For example, the call of line 19 of $S_{\pi\text{-ADL-Spec}}$ means that the transition of S_{IOSTS} labeled with *Coin?* should stay in the same location, while the call of line 22 signifies that the transition labeled with *Return!* should go to p_1.

The composition of two components (abstractions) is modeled by the parallel composition between two IOSTS with synchronization on the actions, which should communicate together. The architectural specification of the coffee machine is the result of the composition between two IOSTS (see Fig.5 and Fig.6) used to model behaviors of the payment and beverage components of the coffee machine. This specification is used in order to derive test cases, however we did not show it in the paper due to its size (20 locations and about 70 transitions).

E. Symbolic Test Generation

Symbolic Test Generation consists in computing, from the formal specification of a system under test and from a test purpose describing a set of behaviors to be tested, a reactive program, called a test case, that observes an implementation of the system to detect non-conformant behavior, while trying to control the implementation towards satisfying the test purpose. The STG tool [17], [18], used for test case generation, takes as inputs an IOSTS specification and an IOSTS test purpose, and then it produces an IOSTS test case. In Section IV-D, we described how to obtain the IOSTS specification from the one written in the π-ADL-Spec language. Bellow we explain the notions of test purpose and test case.

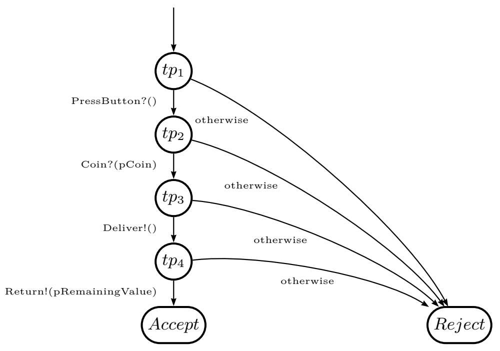


Fig. 10. The test purpose represented by an IOSTS.

1) Test purpose. A test purpose is used to select the behaviors from the specification that are to be exercised by the derived test. Fig.10 illustrates a test purpose that selects from the coffee machine specification a test case that exercises a coffee delivery in the case where the beverage button is pressed and a single coin, which should be sufficient for a coffee payment, is inserted into the coffee machine.

The generation of test cases takes place through the computation of the product between the specification IOSTS and the test purpose IOSTS. Thus, locations in the test case are pairs made up of a location from the specification and a location from the test purpose, and transitions between these locations are added when (1) a specification transition action has the same label as a test purpose action, or (2) the specification is capable of advancing on an internal action. The locations "Accept" and "Reject" in the test purpose indicate locations in the test case that should be interpreted as final. The location "Accept" indicates a successful execution of the tests, while the location "Reject" indicates the behavior of the coffee machine specification in which we are not interested for the moment.

The test purpose of Fig.10 was constructed to select a behavior that (1) begins with the *PressButton?()* action, (2) waits for a coin (see *Coin?(pCoin)*), and then (3) delivers a coffee through the *Deliver!()* action, and (4) returns the rest of amount that has been paid (see *Return!(pRemainingValue)*). Note that, we are not interested in testing behaviors of the coffee machine canceling a command. That is why the*Cancel* action leads to the "Reject" location. For the sake of simplicity, all the arrows of Fig.10 leading to "Reject" are labelled with *otherwise*. This indicates that we are not interested in all the actions except of the authorized ones. For example, in the location $p_1b_1\_tp_1$ the authorized action is *PressButton?()*, all the others, i.e., *Cancel?()*, *Coin?(pCoin)*, *Deliver?()*, and *Return?(pReminingValue)*, go to the "Reject" location.

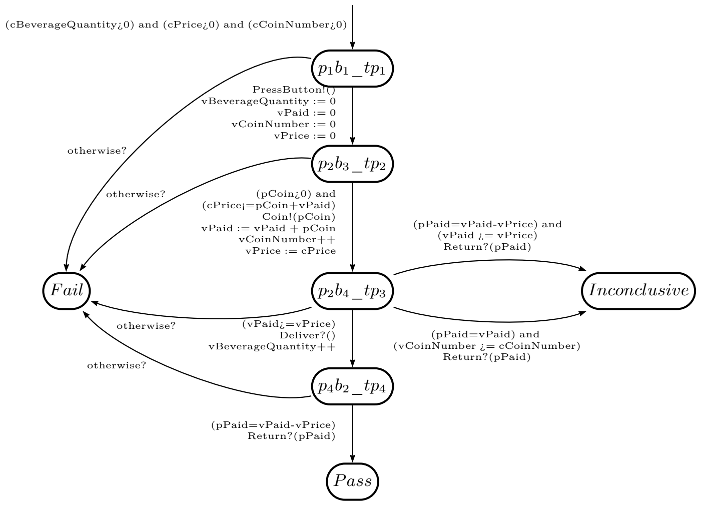


Fig. 11. The test case represented by an IOSTS.

2) Test case. Finally, Fig.11 shows the IOSTS that results from the symbolic test generation using the architectural specification of the coffee machine and the test purpose of Fig.10. Note that, this test case is specific to the test purpose indicated above. Different test purposes will generate different tests. The computation steps carried out are identical to those given in the specification. Actions have had their orientation (i.e., input vs. output) reversed so that the test case becomes a generator of commands and a receiver of responses, complementary to an implementation of the specification. The location labeled "Pass" in Fig.11 indicates that a correct interaction between the tester and the system under test took place. The symbolic test generation method also generates transitions from every location to a new location "Fail" that absorbs incorrect responses from the system under test and lead to the "Fail" state, indicating the non-conformance of the implementation. For each possible erroneous input action received by the tester, the test case generates a transition to "Fail" labeled, for the sake of clarity of the presentation, with the *otherwise?* action from each location of the graph. Note that, the test shown on Fig.11, like all the tests generated by this method, incorporates its own oracle. All of the computation steps necessary to verify the correctness of numeric results are extracted from the specification and used by the tester to verify arguments as they are received. This is in contrast to test generation techniques that simply produce a sequence of inputs to drive the implementation through a specific path.

F. From Abstract to Executable Test Case

In this section, we explain how an abstract test case represented by an IOSTS is translated into an executable code to be run on the black-box implementation of a system under test. First of all, the test case, shown in Fig.11 and called TC_{IOSTS}, is translated into the π-ADL-C&C component, shown on Fig.12 and called $TC_{\pi\text{-ADL-C\&C}}$, as follows:

a) The symbolic constants of TC_{IOSTS}, such as *cCoinNumber*, *cBeverageQuantity*, and *cPrice*, are transformed into parameters of $TC_{\pi\text{-ADL-C\&C}}$ (see lines 2-4).

```
 1 component TestCase is abstraction(
 2    cCoinNumber : Natural, // 10
 3    cBeverageQuantity : Natural, // 15
 4    cPrice : Natural) // 2
 5
 6    port is {
 7       connection Coin is out (Natural).
 8       connection Cancel is out ().
 9       connection PressButton is out().
10       connection Return is in (Natural).
11       connection Deliver is in().
12    }.
13    ...
14    P2B3_TP2 is abstraction(
15       vBeverageQuantity : location[Natural],
16       vPaid : location[Natural],
17       vCoinNumber : location[Natural],
18       vPrice : location[Natural]
19    ){
20       choose {
21          pCoin : location(4).
22          if ((cPrice' <= pCoin'+vPaid') and (pCoin' > 0)) then{
23             via Coin send pCoin.
24             vPaid := vPaid'+pCoin.
25             vCoinNumber := vCoinNumber'+1.
26             vPrice := cPrice'.
27             P2B4_TP3(vBeverageQuantity',vPaid',vCoinNumber',vPrice')
28          }
29       or
30          via Deliver receive.
31          Fail()
32       or
33          via Return receive pPaid : location[Natural].
34          Fail()
35       }
36    }
37    P2B4_TP3 is abstraction(
38       vBeverageQuantity : location[Natural],
39       vPaid : location[Natural],
40       vCoinNumber : location[Natural],
41       vPrice : location[Natural]
42    ){
43       choose {
44          via Deliver receive.
45          if (vPaid'>=vPrice') then{
46             vBeverageQuantity := vBeverageQuantity'+1.
47             P4B2_TP4(vBeverageQuantity',vPaid',vCoinNumber',vPrice')
48          }else{ Fail() }
49       or
50          via Return receive pPaid : location[Natural].
51          if ((pPaid'=vPaid'-vPrice') and (vPaid'>=vPrice')) then{
52             Inconclusive()
53          }else{ Fail() }
54       or
55          via Return receive pPaid : location[Natural].
56          if ((pPaid'=vPaid') and (vCoinNumber'>=cCoinNumber')) then{
57             Inconclusive()
58          }else{ Fail() }
59    }
60    ...
61    Pass is abstraction(){ print("PASS") }
62    ...
63    behaviour is { P1B1_TP1(0,0,0,0) }
64 }
```

Fig. 12. The extract of the π-ADL C&C test case.

b) The input/output actions of TC_{IOSTS} (*Deliver?*, *Return?*, and *Coin!*, *Cancel!*, *PressButton!*) play the role of connectors in $TC_{\pi\text{-ADL-C\&C}}$ (see lines 7-11).

c) Each location of TC_{IOSTS} is transformed into an abstraction of $TC_{\pi\text{-ADL-C\&C}}$. All the abstractions, except the ones corresponding to the test verdicts, have the same number of parameters. These parameters correspond to the variables of TC_{IOSTS}. For example, the location $p_2b_3\_tp_2$ of TC_{IOSTS} is translated into the abstraction P2B3_TP2 (see lines 14-36), which has four parameters: *vBeverageQuantity*, *vPaid*, *vCoinNumber*, and *vPrice*. Notice that, the special locations, such as *Pass*, *Fail*, and *Inconclusive*, correspond to the abstractions without parameters (e.g., the location *Pass* corresponds to the abstraction represented by the code on line 61). The role of these abstractions is to produce a test verdict.

d) For each location of TC_{IOSTS}, each outgoing transition is translated into one case of the "**choose**" structure of the abstraction corresponding to this location. For example, the transition t_1 with origin $p_2b_3\_tp_2$ and destination $p_2b_4\_tp_3$ labeled with the *Coin!(pCoin)* output action corresponds to the first case of the "**choose**" structure of P2B3_TP2 (see lines 21-28). Notice that, the destination of t_1 is modeled by a call to the P2B4_TP3 abstraction. The code, corresponding to a guarded transition labeled with an output action, is surrounded by the "**if(...)then**{...}" structure, where the guard of this transition appears as a condition. Moreover, in order to fire a transition labeled with an output action carrying parameters, a test case should automatically generate values for these parameters satisfying the guard of this transition if it is present. At the moment, such parameters are instantiated with values chosen by the test developer. For example, the *pCoin* parameter is instantiated with 4. This value satisfies the guard of the transition t_1, i.e., $(pCoin > 0)$ and $(cPrice \leq pCoin + vPaid)$ if the price of the beverage is 3, for example. The code, corresponding to a guarded transition labeled with an input action, is surrounded by the "**if(...)then**{...}**else**{...}" structure, where the guard of this transition appears as a condition. The input action should be invoked just before this structure as we need to know received values of its parameters. Notice that, if the guard/condition is not satisfied, then the test case generates the "Fail" verdict. For example, the code corresponding to lines 44-48, models two transitions of TC_{IOSTS} outgoing from the $p_2b_4\_tp_3$ location and labeled with the *Delivery?()* action. One of them permits to reach the $p_4b_2\_tp_4$ location, if the guard $g : vPaid \geq pPrice$ is satisfied, and other goes to the "Fail" location, if the guard g is unsatisfied.

e) The behavior of the test case $TC_{\pi\text{-ADL-C\&C}}$ is modeled by a call to the P1B1_TP1 abstraction, which corresponds to the initial location of TC_{IOSTS}.

To obtain an executable test case, a test case expressed in the π-ADL-C&C language is automatically translated into π-ADL-Spec code (see Section IV-A), and then into a concrete executable test program expressed in the π-ADL.NET language (see Section IV-B).

G. Test Case Execution

The last step of our approach is to compile and to execute the π-ADL.NET test case obtained from an abstract test case, represented by IOSTS, as was explained in Section IV-F. This test case is executed on a real black-box implementation of the system under development, where the execution is modeled by the parallel composition between the test case and the implementation with synchronization on common input/output actions. The results of a test execution are: "Pass", meaning no errors were detected and the test purpose was satisfied, "Inconclusive" – no errors were detected but the test purpose was not satisfied, or "Fail" – the implementation exhibits a non-conformance with respect to the architectural specification in a behavior targeted by the test purpose.

V. Tool Support

A major impetus behind developing formal languages for architectural description is that their formality renders them suitable to be manipulated by software tools. The usefulness of an ADL is thereby directly related to the kinds of tools it provides to support architectural description, but also analysis, refinement, code generation, and evolution. Indeed, we have developed a comprehensive toolset for supporting architecture-centric formal development around π-ADL. It is composed of:

– a *callable compiler* and a persistent *virtual machine* for executing architecture descriptions based on the operational

semantics of π-ADL (implemented in C# on the .NET platform) [20];
– three transformators implemented in C++ and allowing to translate (1) a π-ADL-C&C code into a π-ADL-Spec code, (2) a π-ADL-Spec code into a π-ADL.NET code, and (3) a π-ADL-Spec code into an IOSTS model.
– a π-ADL-C&C syntax checker implemented in C++.

The work presented in this paper adds a new method and tool for architecture validation based on conformance testing. Indeed, in order to validate the conformance of the executable system with respect to its architectural specification, we apply the conformance testing technique, i.e., tests are generated automatically, using the STG tool [17]–[19], and then they are executed on the system under test. To be able to generate tests from a π-ADL-Spec architectural specification with STG, the specification should be translated into a low-level IOSTS model. This step is almost automatized. The STG tool generates abstract test cases expressed by IOSTS, therefore we need also to transform them into the π-ADL-C&C language (this step is done manually, at the moment).

VI. Summary and Related Work

The main purpose of this paper is to propose an approach that permits (1) to easily design the architecture of a system under development using π-ADL, (2) to automatically generate an implementation of this system that can be executed on the platform .NET, and (3) to test the conformance of the implemented system against its architectural specification.

As it is mentioned in [2] and [3], several works propose different formal and semi-formal ADLs for the description of software architecture. Some of these ADLs rely on Labeled Transition Systems (LTSs) used to model the behaviors of a software architecture, for example, Chemical Abstract Machine (CHAM) [21], Finite State Process (FSP) [22], and π-ADL [4]. As this paper is based on our previous work [4], the choice of π-ADL, as a language for architecture design, is natural for us. Once the architecture of a software system is designed using the user-friendly π-ADL-C&C language, we refine it into a low-level π-ADL.NET architecture that can be compiled, using the compiler [20] developed in our research team, and executed on the .NET platform.

The choice of π-ADL allows us to use formal methods in order to assure the quality of a system under development. Indeed, π-ADL is a formal, well theoretically founded ADL. Moreover, the behaviors of designed systems can be captured by means of transition systems. In this work, we use a testing technique in order to check the conformance of a system's implementation with respect to its architectural specification, and therefore to assure the quality of this system. This work is based on our previously proposed technique [19] and on a tool [17], [18] allowing automatic test generation for reactive programs (written in Java or C++) from low-level specifications modeled by IOSTSs.

The closest works to our proposal are these of Muccini, Bertolino, and Inverardi [11], [12], [23], [24]. Their approach consists of the automatic derivation of suitable abstract test cases from the behaviors of a system under test that is modeled by LTS. The test cases are selected by the use of Abstract LTS (ALTS) allowing to abstract away uninteresting, for the moment, system's actions, and then applying the coverage criterion of McCabe (another criteria can also be used) to obtained abstract test cases. One of the difficulties of this approach underlined by the authors, is to establish a relationship between the system at its abstract architectural level and the system's implementation. It is needed in order to obtain concrete executable test cases from the abstract ones.

In the approach presented in this paper, we generate abstract test cases from the IOSTS model of an architectural specification written in π-ADL. We use the notion of test purpose, as a test selection mechanism, in order to focus on specific behaviors of the system under test. The inconvenient is that we do not generate the test purposes automatically, therefore their elaboration needs a human intervention. On the other hand, the translation of abstract test cases is quite straightforward in our approach as it was described in Section IV.

Bellow we listed other related works that have been done in the domain of architectural testing. This list is certainly not exhaustive. The authors of [6] define six architectural-based testing criteria and use them in order to generate test plans from the software architecture modeled by CHAM by adapting existing specification-based techniques to the domain of architecture-based testing. In [7], Bertolino and Inverardi use the architectural testing in order to test extra-functional properties of a system under test. Tracz [25] shows how to use Domain-Specific Software Architecture (DSSA) in order to capture structural and temporal properties of a system under development. He gives some ideas on how architectures can be specified to enable its analysis and testing. In [8], [26], the authors propose dependence analysis techniques based on software architecture and called chaining. In [27], Rosenblum adapts its component-based test strategy based to an architecture-based test of software systems. This approach is based on the architectural models that can be simulated, executed, or used to realize the integration or regression testing on the implementation of a system under test. Finally, the author describes how formal models, combined with architectural models, can be used to guide software testing. In [9], Harrold presents approaches for using software architecture for effective regression testing. In [28], she also discusses the use of software architecture for testing. In [10], the authors define several test criteria, and propose techniques, and automated tools for the specification and generation of system level tests from architectural descriptions. Muccini and his colleagues are also interested by regression testing. Their contribution to this topic can be found in [13], [29], [30]. These works explore the question how regression testing can be systematically applied to the software architecture to reduce the cost of regeneration tests for modified systems. The authors are interested in two types of changes of a software system, which are (1) modification of the architecture and (2) modification of the implementation. There is also an interesting work of Bertolino [31] discussing different important achievements in the field of software testing and listing the most relevant challenges to be addressed in this field.

VII. Conclusion

This paper has presented a formal approach which, starting from the architecture of a software system, generates a system implementation and tests it at the architectural level. In particular, this approach has been applied to software systems designed using high-level architecture description language called π-ADL. The test part of the approach is based on symbolic test generation, which (1) automatically derives test cases in order to check the conformance of a system with respect to the behavior of an architectural specification selected by the test purposes; (2) automatically determines whether the results of the test execution are correct with respect to the architectural specification. It performs test derivation as a symbolic process, up to and including the generation of test program source code. The reason to use symbolic techniques instead of enumerative is that symbolic test generation allows us to produce (1) more general test cases with parameters and variables, which should be instantiated only before the test cases execution, and (2) test cases that are more readable by humans. We validated our approach on a simple example of the coffee machine.

As it was mentioned in this paper, some steps of our approach are semi-automatized, therefore, the first direction of our future work is to render the approach completely automatic from test generation down to test execution. To show the feasibility and utility of our approach we plan to apply it to a realistic case study. Second, we plan to work on the implementation of a mechanism to automatically compute test purposes from the system architectural specification using, for example, coverage criteria instead of test purposes written by hand. Third, we plan to extend our approach by incorporating in it a technique of model checking in order to enable the automatic verification of critical parts of a system under development.

References

[1] S. Shaw and D. Garlan, *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.

[2] N. Medvidovic and R. N. Taylor, "A classification and comparison framework for software architecture description languages," *IEEE Trans. on Software Eng.*, vol. 26, no. 1, pp. 70–93, 2000.

[3] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang, "What industry needs from architectural languages: A survey," *IEEE Trans. Software Eng.*, vol. 39, no. 6, pp. 869–891, 2013.

[4] F. Oquendo, "π-adl: an architecture description language based on the higher-order typed pi-calculus for specifying dynamic and mobile software architectures," *SIGSOFT Softw. Eng. Notes*, vol. 29, no. 3, pp. 1–14, 2004.

[5] G. Myers, *The Art of Software Testing*. John Wiley & Sons, 1979.

[6] D. J. Richardson and A. L. Wolf, "Software testing at the architectural level," in *Proc. of ISAW and Viewpoints'96 on SIGSOFT'96 workshops*, ser. ISAW'96. New York, NY, USA: ACM, 1996, pp. 68–71.

[7] A. Bertolino and P. Inverardi, "Architecture-based software testing," in *Proc. of ISAW-2 and Viewpoints'96 on SIGSOFT '96 workshops*, ser. ISAW'96. New York, NY, USA: ACM, 1996, pp. 62–64.

[8] J. A. Stafford, D. J. Richardson, and A. L. Wolf, "Chaining: A software architecture dependence analysis technique," 1997.

[9] M. J. Harrold, "Architecture-based regression testing of evolving systems," in *Proc. of the Int. Workshop on the Role of Software Architecture In Testing and Analysis*, ser. ROSATEA'98, 1998, pp. 73–77.

[10] Z. Jin and J. Offutt, "Deriving tests from software architectures," in *Proc. of the IEEE Int. Symposium on Software Reliability Engineering*, ser. ICSE'01, 2001, pp. 308–313.

[11] A. Bertolino, P. Inverardi, and H. Muccini, "Formal methods in testing software architectures," in *SFM*, 2003, pp. 122–147.

[12] H. Muccini, A. Bertolino, and P. Inverardi, "Using software architecture for code testing," *IEEE Trans. on Software Engineering*, vol. 30, no. 3, pp. 160–171, March 2004.

[13] H. Muccini, M. S. Dias, and D. J. Richardson, "Reasoning about software architecture-based regression testing through a case study," in *Proc. of the Computer Software and Applications Conf.*, ser. COMPSAC'05, 2005, pp. 189–195.

[14] B. Beizer, *Software Testing Techniques*. New York: Van Nostrand Reinhold, 1990.

[15] G. J. Tretmans, "A formal approach to conformance testing," Ph.D. dissertation, University of Twente, the Netherlands, December 1992.

[16] D. Sangiorgi, "Expressing mobility in process algebras: First-order and higher-order paradigms," Ph.D. dissertation, University Edinburgh, UK, February 1992.

[17] D. Clarke, T. Jéron, V. Rusu, and E. Zinovieva, "STG: A Symbolic Test Generation tool," in *Proc. of the 8th Int. Conf. on Tools and Algorithms for the Construction and Analysis of System (TACAS'02)*, ser. LNCS, vol. 2280, Grenoble, France, April 2002, pp. 470–475.

[18] F. Ployette and F.-X. Ponscarme, "The STG tool page," Available at http://www.irisa.fr/prive/ployette/stg-doc/stg-web.html, October 18, 2007.

[19] E. Zinovieva-Leroux, "Symbolic methods in test generation for reactive systems with data," Ph.D. dissertation, University of Rennes 1, France, November 22, 2004.

[20] Z. Qayyum and F. Oquendo, "The π-adl.net project: an inclusive approach to adl compiler design," *WSEAS Transactions on Computers*, vol. 7, no. 5, pp. 414–423, May 2008.

[21] P. Inverardi and A. L. Wolf, "Formal specification and analysis of software architectures using the chemical abstract machine model," *IEEE Trans. on Software Eng.*, vol. 21, no. 4, pp. 373–386, 1995.

[22] J. Magee, J. Kramer, R. Chatley, S. Uchitel, and H. Foster, "Ltsa - labelled transition system analyser," Available at http://www.doc.ic.ac.uk/ltsa/, June 04, 2009.

[23] A. Bertolino, F. Corradini, P. Inverardi, and H. Muccini, "Deriving test plans from architectural descriptions," in *Proc. of the 22nd Int. Conf. on Software Engineering*, ser. ICSE'00. New York, NY, USA: ACM, 2000, pp. 220–229.

[24] A. Bertolino, P. Inverardi, and H. Muccini, "An explorative journey from architectural tests definition downto code tets execution," in *Proc. of IEEE Int. Symposium on Software Reliability Engineering*, ser. ICSE'01, 2001, pp. 211–220.

[25] W. Tracz, "Test and analysis of software architectures," in *Proc. of the 1996 ACM SIGSOFT Int. Symposium on Software Testing and Analysis*, ser. ISSTA'96. New York, NY, USA: ACM, 1996, pp. 1–3.

[26] J. Stafford, D. Richardson, and A. Wolf, "Aladdin: A tool for architecture-level dependence analysis of software systems," University of Colorado, Tech. Rep. CU-CS-858-98, 1998.

[27] D. Rosenblum, "Challenges in exploiting architectural models for software testing," in *Proc. of the Int. Workshop on the Role of Software Architecture in Testing and Analysis*, ser. ROSATEA'98, Italy, Jul. 1998, pp. 49–53.

[28] M. J. Harrold, "Testing: a roadmap," in *Proc. of the Conf. on The Future of Software Engineering*, ser. ICSE'00. New York, NY, USA: ACM, 2000, pp. 61–72.

[29] H. Muccini, M. Dias, and D.Richardson, "Towards software architecture-based regression testing," in *Workshop on Architecting Dependable Systems (WADS)*, ser. ICSE'05, vol. 30:4. St. Louis, Missouri (USA): ACM, May 2005, pp. 1–7.

[30] ——, "Towards software architecture-based regression testing," University of L'Aquila, Tech. Rep., 2005.

[31] A. Bertolino, "Software testing research: Achievements, challenges, dreams," in *Proc. of the Future of Software Engineering*, ser. ICSE'07. IEEE-CS Press, 2007, pp. 85–103.

Combining Model-based Testing and Continuous Integration

Martin Koskinen, Dragos Truscan, Tanwir Ahmad and Niklas Grönblom
Åbo Akademi University,
Turku, Finland

Email: [martin.koskinen, dragos.truscan, tanwir.ahmad, niklas.gronblom]@abo.fi

***Abstract*—We present our approach on combining model-based testing with the continuous integration process. The main benefits of this combination lie in the ability to automatically check the conformance of the implementation with respect to its specification, while shortening the feedback cycles and providing increased test coverage. A case study on developing an in-house academic tool is presented in which the online model-based testing approach is used with the continuous integration process.**

Keywords*—*Continuous Integration; Model-Based Testing; UPPAAL Timed Automata

I. Introduction

Continuous Integration (CI) is a software development practice in which developers frequently integrate their work [1] [2]. The CI process runs continuously during the lifetime of a project, resulting in that the different parts of the product are always updated, integrated, and tested. Regression test suites are used for checking the quality of the integration.

One perceived problem of CI is that the increasing size of the code base leads to increasing run time of the integration build. According to Rogers [3], one of the main causes behind this, is not the increasing compilation time, but rather the increasing number of tests executed. In addition, the maintenance of the regression test suites can be time consuming and error-prone. As current practice, Duvall et al. [2] recommend that system builds are run several times, or at least once a day, which imposes tight constraints on the CI and testing process.

In this paper, we discuss the inclusion of the model-based specifications and automated test design techniques into the CI process, in order to enable incremental development, shorter feedback cycles and increased test coverage. There are two enablers for achieving these targets: (a) early detection of errors is facilitated by performing simulation and verification on the model-based specification after each update of the specifications; and (b) the test suite corresponding to the latest version of the specifications is generated automatically and made available to the CI process.

Our testing approach focuses on the conformance testing using automated test generation techniques. The system is developed incrementally. The specifications are done using *UPPAAL timed automata* (UPTA) [4]. Every time a new feature is added, it is first modeled, simulated and verified. Once the specifications are updated, they are used for automated test generation. When the feature is also implemented in the source code, the automatically generated tests are executed in order to detect possible behavioral inconsistencies between the specification and the implementation, and a report is issued as feedback.

The paper has the following structure: Section II will briefly discuss different background concepts. In Section III we introduce a generic process for combining MBT with CI, followed by a concrete case study in Section IV. Section IV also describes how we applied this approach in a practical software development project. An evaluation of our approach is discussed in Section V, whereas final thoughts and future work are presented in Section VI.

II. Background

In the following subsection, the CI process is described in more detail, followed by a short introduction to *Model-based testing* (MBT) [5]. The last subsection briefly introduces the UPPAAL tool and its capabilities.

A. Continuous Integration revisited

The traditional workflow of the CI process can be summarized as following. When a developer has finished an implementation task, he makes a local build to see whether the program builds correctly. Ideally, he also runs tests locally to verify that the implementation is correct. After this, the developer commits the code to the *Source Code Management* (SCM) system.

A CI-server is used to integrate source code from different SCMs used in the process and to create an integration *build*, either at regular time intervals or based on commit triggers linked to the SCMs. The build process might contain different kinds of code analysis, for example to ensure that the code conforms to common code conventions, or integration/acceptance tests for the newly built software. Subsequently, feedback is provided to the concerned parties on the outcome of the build. If errors occurred or conventions were violated, these are mentioned in the report. When errors or violations are detected in the integration build, the responsible developer is supposed to fix them as soon as possible. In many instances of CI, the CI process is stopped, i.e., no one can commit updates, before the previous failed build is fixed or the code is reverted to the previous working version.

B. Model-based Testing

MBT is a testing approach which reduces the effort needed for testing [6], by automatically designing test suites from abstract behavioral specifications of the *system under test* (SUT). The main philosophy behind MBT is to automatically generate tests from abstract models, which specify the expected behavior of the system under test. Based on how tests are generated and executed MBT has two flavors: online and offline [7]. In *online* testing, tests are generated from the model and executed on-the-fly against the SUT. At each step, a new test is designed based on the output of the previous test. In contrast, in *offline* mode, all test are pre-generated (scripted) into an executable format, which is then executed in batch mode using test execution frameworks.

C. UPPAAL

UPPAAL is a toolbox for verification of real-time systems [4]. The tool provides a graphical user interface for editing, simulating and verifying models based on an extended version of time automata, referred to as UPTA [4].

Informally, in UPTA, systems are modelled as a network of timed automata which communicate with each other through global variables and channel synchronizations. An edge, that connects locations, can be decorated by a guard, allowing or not allowing the edge to be taken, depending on some condition. A channel can be sending or receiving synchronizations, which are annotated by the suffixes *!* and *?* subsequently. An edge with a sending synchronization requires one edge that can receive the synchronization. If several receiving channels are available, one will be chosen non-deterministically. Synchronization channels can be declared as broadcast channels, which removes the requirement of a synchronization receiver. This implies that broadcast synchronizations will be sent, even if there is no receiver. If there are several receivers available for a broadcast, all receivers will receive the synchronization simultaneously. On edges, variable values are updated by assignment statements. Initial locations are marked with a double circle. There are two other special location types except the normal location. An urgent location, which stops time, is marked by an 'U'. A committed location, marked by a 'C', is stricter than the urgent one, since the automaton is allowed to leave the location in the next transition without intervention by another process. For a formal definition of UPTA, one could refer to [4].

III. Combining MBT and CI

As mentioned in the introduction, in order to take advantage of the MBT approach, we integrate MBT into the CI process. We use MBT to make sure that the specification and the implementation of the SUT always conform to each other. A generic view of our CI process is shown in Figure 1.

The CI process employs several SCM servers, used for maintaining different artifacts of the development process. In Figure 1, there are different SCMs for versioning the source code, specifications, toolchain, test suites, etc. In practice, one can use the same SCM server for accommodating several or all artifacts.

Several teams are involved in the development, for instance, a specification and a development team, each following specific processes and committing regularly (with different frequency) to the corresponding SCM server. Basically, when a new feature is introduced, it is specified, validated and then committed. Validation helps in detecting potential inconsistencies in the specifications, such as misunderstanding of requirements or omissions. The simulation and validation ensure that the desired behavior can be achieved.

The task of the development team is to implement the requested features according to the specifications. The developers can test their code locally, after which they commit it to the corresponding SCM server. These tests may be unit tests developed by the developers themselves or tests retrieved from the test suite SCM.

As the main idea of software testing is to verify the behavioral conformance between the specification and the implementation, every time one of them is updated, we check that they conform to each other using MBT.

This process is controlled by a CI server, which is configured to monitor the SCM repositories involved in the build. Whenever a commit to any of the repositories is detected, all repositories related to the project are updated on the CI server. Regardless of which repository triggered the update, the following steps are executed depending whether an offline or online testing approach is used.

a) Offline process: When a build is triggered due to changes on a SCM server storing the specifications, the CI server checks if the existing test suite needs updating due to changes to the specifications. If needed, a fresh test suite is generated from scratch. The test generation replaces the need for maintaining and deciding which tests should be added or removed from a manually created regression test suite. The updated test suite is stored on a SCM server for later use. The test suite is generated once and can be reused as long as the specifications do not change. Updating the test suite can be done as part of the CI, or as a separate process as for convenience is showed in Figure 1.

When the code is updated on the corresponding SCM server, a build is started. Upon completion, the test suite is executed in batch mode against the SUT. Finally, the developers get feedback on how the build proceeded.

b) Online process: For distinction, the online testing process is depicted with thicker line pattern in Figure 1. When a build is triggered by either of the SCM servers, the CI server starts by building the project. As in online mode the tests are generated and executed on-the-fly, there is no previous test suite as such to be updated.

IV. Case Study

In this section, we provide a concrete example of how the generic process described in Section III can be put into practice. The case study presented in this paper is part of the development process of an academic tool for performance

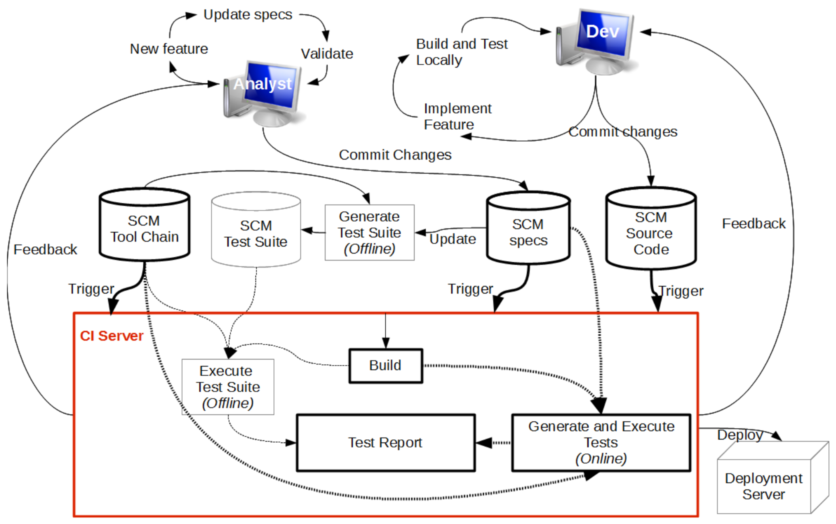


Figure 1. Generic process overview

testing, called MBPeT. MBPeT [8] is a model-based load generation tool which uses probabilistic models to generate load and applies it interactively against the target system. The development team consists of 2-4 developers, while the specification team consists of 1-3 persons.

In the following subsections, we briefly describe different activities involved in our development process.

A. *Model-based specifications*

The MBPeT tool has a distributed architecture, in which one master node controls several slave nodes that are actually generating the load by executing the desired number of concurrent virtual users. During load generation, the master node decides how the load is distributed to the slaves. Each slave has a predefined saturation threshold for the local resources which is used to ensure that the slave node is able to generate the required load. Whenever the saturation threshold is reached, the load on the current slave is kept constant, while the remaining load is delegated to the next available slave.

In the specification phase, the models of both the master node (see Figure 2) and the slave node (see Figure 3) are created and their communication is modelled in UPTA. The behavior of the two node types is described in the following:

c) Master model: The master process in Figure 2 is designed to handle several slaves. It starts by waiting for all slave processes to connect, by receiving the *i_slave_connect* synchronization from each slave process. Then the master process continues with configuration and initialization of the connected slaves, by sending an *o_initSlave* synchronization to each slave process. The initialization is completed when all slave processes have sent an *i_slaveInit* synchronization to the master. At this point, the master requests the first slave to start load-generation by sending an *o_generateLoad* synchronization to it. The master continues to listen for either an *i_slaveSaturated* or an *i_slaveDone* synchronization. If an *i_slaveSaturated* is received and there are no available slaves to start, the `failure` variable is set to *1*. When the master process has received the *i_slaveDone* synchronization from all started slaves, the master proceeds by shutting down all connected slaves, by sending an *o_killSlave* synchronization to each slave process. At the end of the test session, the master process enters the *STOPPED* location.

d) Slave model: The process model, corresponding to the previously described master process, is shown in Figure 3. The process starts by sending an *i_slave_connected* signal to the master process. The master initializes the slaves by sending configuration information, which corresponds to the slave process reaching the *Initialized* location. Here it waits for either an *o_killSlave* or an *o_generateLoad* synchronization. The former results in a return to the initial location and the latter instructs the slave process to start generating load. The slave saturation is calculated by looping via the locations *Load_calculated* - *Saturation_Check*. If the slave's load variable is greater than a *threshold* value, the slave is considered saturated. When the slave becomes saturated, it transitions to the *generate_load_saturated* location by sending an *i_slaveSaturated* synchronization. When test time runs out, the slave transitions to the initial location via the *Load_generation_completed* location. If the test duration runs out without the slave being saturated, the slave transitions to the *Load_generation_completed* location via location *TestDuration_timeout* and sends an *i_slaveDone* synchronization to the master. The slave instances share a global clock *timer*, which is used for exiting the load generation when test time runs out. The clock is reset when the first slave starts generating load. For the rest of the slaves, the load generation is started without resetting the timer. The models discussed above allow for an instance of the master to communicate with several instances of the slaves, thus imitating the architecture of the real tool.

Simulation The models of the MBPeT tool have been created incrementally, one feature being added at the time.

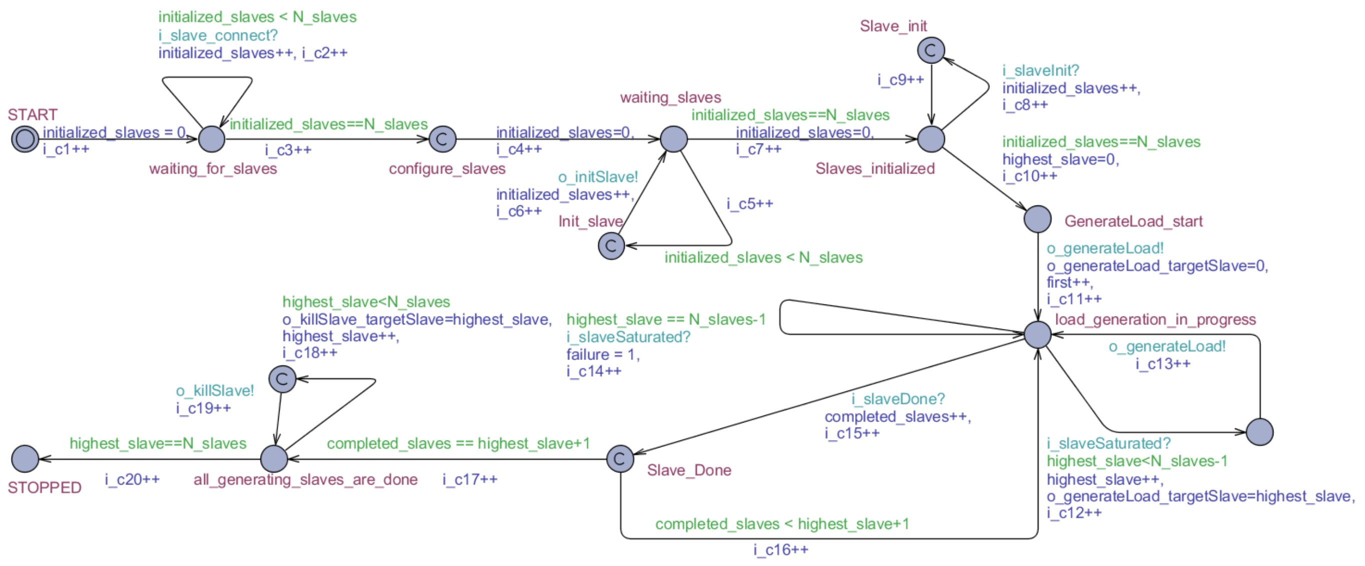
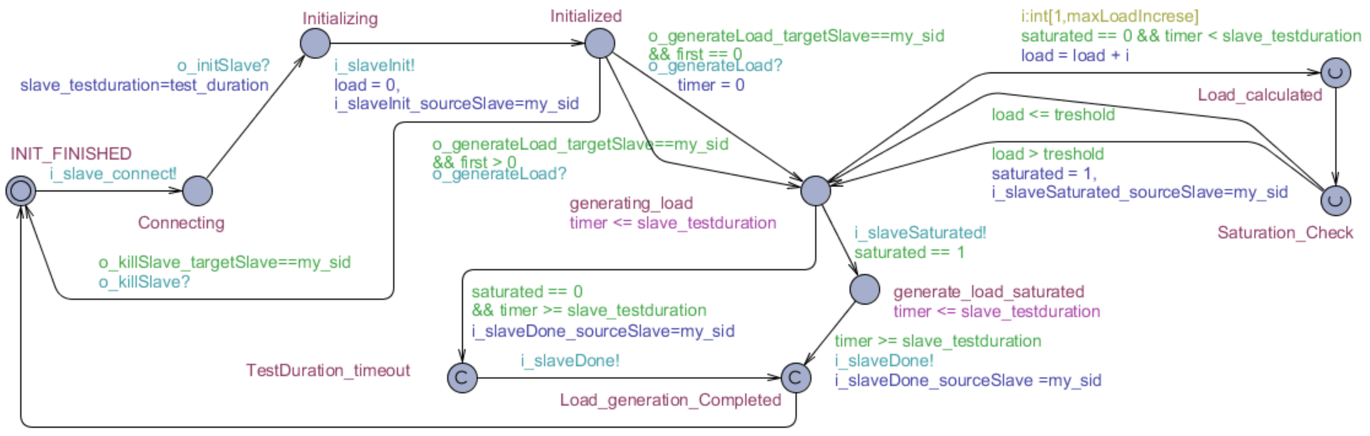


Figure 2. Master process model



Initializing
Initialized
o_initSlave?
slave_testduration=test_duration
i_slaveInit!
load = 0,
i_slaveInit_sourceSlave=my_sid
o_generateLoad_targetSlave==my_sid
&& first == 0
o_generateLoad?
timer = 0
i:int[1,maxLoadIncrese]
saturated == 0 && timer < slave_testduration
load = load + i
Load_calculated
load <= treshold
load > treshold
saturated = 1,
i_slaveSaturated_sourceSlave=my_sid
Saturation_Check
INIT_FINISHED
i_slave_connect!
Connecting
o_generateLoad_targetSlave==my_sid
&& first > 0
o_generateLoad?
generating_load
timer <= slave_testduration
o_killSlave_targetSlave==my_sid
o_killSlave?
i_slaveSaturated!
saturated == 1
generate_load_saturated
timer <= slave_testduration
saturated == 0
&& timer >= slave_testduration
i_slaveDone_sourceSlave=my_sid
TestDuration_timeout
i_slaveDone!
timer >= slave_testduration
i_slaveDone!
i_slaveDone_sourceSlave =my_sid
Load_generation_Completed

Figure 3. Slave process model

After each feature is modeled, the models are simulated in the UPPAAL tool, allowing us to experiment and check if the proposed models behave as specified in the requirements.

Verification The models are also used for verification of different system properties, e.g., reachability, safety, and liveness. UPPAAL provides its own verification engine [4], which uses a simplified version of the TCTL language [9]. For brevity, we refer the reader to [4] for more details.

B. CI process

In our development process, we have used three repositories: one for the implementation of the SUT, one for the specifications of the SUT and one for the tool chain. We follow the online testing process described in Section III.

1). When a new feature is to be introduced to the MBPeT tool, it is first modelled in UPTA. The specification is simulated and verified. When the new feature is considered "approved", the new models are submitted to the specifications SCM. The first time, this commit will trigger a build which will fail, since the specification contain an unimplemented feature. The failed build shows that the system implementation is lagging behind the specifications, i.e., it does not conform to the specified behavior.

2). The development team will start implementing the new feature in code. When the implementation is ready, the developers run unit tests locally. The development team also has access to the entire testing tool chain to validate their updated implementation before committing. When the code is committed, a new build is started followed by the model-based testing of the build. If the integrated system behaves according to the specification, the build will pass the testing successfully; otherwise an error-report will be generated.

3). Both the development and the specification teams will receive an error-report, which will be discussed and analyzed in order to detect the source of the failure(s). It may happen that the error is in the specification, instead of the implementation, due to misunderstanding of the requirements or undetected errors in the specification. The error may also be located in the tool chain or in the test environment.

4). The identified failures are fixed by the team responsible. Upon committing the updated artifact, a new build is triggered which should result in a successful test run.

The CI process is supported by the Jenkins CI server [10], an open source continuous integration server, configurable via a Web interface. Its functionality is extendable via plugins, e.g., integration with SCM systems. The building of projects is configured via *jobs*. A job can be triggered manually, based on a time trigger or based on an event, e.g., the completion of another job.

The SCM software we use is Subversion [11]. In order to implement the job-triggering mechanism, we implemented a set of ”hook” scripts, which are run on the SCM servers and monitor certain paths for commits. When a commit is detected by a hook script, via a regular expression match on the monitored path, the Jenkins CI server job is triggered by an a HTTP request.

C. Test generation

In our approach, we have targeted the online mode of MBT, since it addressed better the non-determinism in the specifications. In our case, study we have non-determinism at several locations, for example in the *generating_load* location. There is no limit on how many times, if any, the loop calculating the slave's saturation is taken.

Since the models of both the master and the slave nodes are created and verified in the specification phase, any one of them can be used as a SUT model, whereas the other one will be used as environment model. Consequently, in our CI environment, we have two independent jobs: one considering the master node as the SUT, while the other uses the slave node as the SUT. In this paper, we selected for exemplification the setup where the master node is the SUT, and the slave nodes act as the test environment.

An overview of our model-based test setup, is shown in Figure 4. We use three repositories: specifications, source code, and tool chain. Whenever one of the repositories is updated the commit trigger is activated and the CI process is started by updating the repositories (if new versions are available), building and deploying the implementation, instantiating the tool chain and starting test execution.

The tool chain is composed of several components. *Distributed TRON* (DTRON) [12] is a tool for distributed online test generation. DTRON uses a Spread network, the Spread Toolkit [13] for its multicast communication between different instances of DTRON and an eventually distributed SUT. In order to interface with the *implementation under test* (IUT), an *adapter* written in Java, is used to convert tests messages received from DTRON into messages compatible with the communication protocols required by the IUT. DTRON will receive output from the SUT via the adapter, which distributes it via the Spread network. The received values are compared with the expected output and a verdict is given.

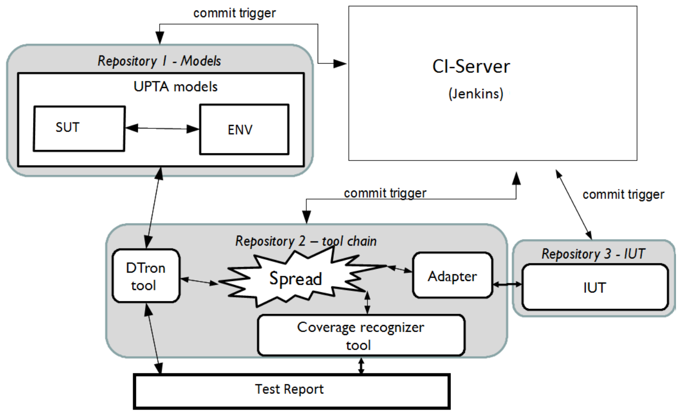


Figure 4. Overview of the test setup

The adapter is updated every time new observable interfaces of the SUT are added to the UPTA specifications. There is a naming convention for making channels and variables observable by DTRON. A channel name prefixed by *o_* means the channel is used for IUT-to-model communication. Similarly, a channel prefixed by *i_* is used for model-to-IUT communication. Integer variables can be sent along with these synchronizations. In this case, the variable name is prefixed by the observable synchronizing channel's name, i.e., *i_channelname_variablename*; see Figures 2 and 3 for exemplification.

Once the test session is started, DTRON will generate tests via symbolic execution of the specifications using randomized choice of input. The observable communication between the environment model and the system model is captured by the adapter and send or expected to be received from the SUT. Whenever an expected output is not received from the IUT with the expected value or within the specified timeframe, a failure will be observed and the test generation and execution will be stopped. Consequently, the build job on the CI-server will generate a test report and will send it to the respective teams.

D. Measuring coverage

In our approach, for each new commit, we measure both specification coverage and code coverage for each test run.

Specification coverage. In order to achieve a certain coverage level, with respect to specific coverage criteria, we have two options available when using UPPAAL-based tools. The first option is to use an environment model which will drive the test generation to follow specific test targets in the SUT model, as described by Hessel et al. [14]. The second option is to have an environment model which does not enforce explicitly specific test targets in the SUT model (e.g., the model of the slave node) and to recognize the coverage level of the test run upon completion.

In our CI process, we use the second approach, which requires an additional utility script to be included in the process. The general idea is to automatically customize the UTPA models, without modifying the original behavior, in order to allow one to observe how different structural parts of the model have been covered during the test execution.

The *coverage recognizer tool* (CRT), as shown Figure 4, has two main functionalities. When the specification is updated and detected by the CI-server, the script processes the UPTA model by adding unique counter variables and a corresponding updates statements on each edge of the SUT model. See for exemplifications variables $i\_c1, \ldots, i\_c20$ added to the Master model in Figure 2.

An observable channel is also added, hereafter referred to as the *counter channel*, which is used for synchronizing the counter variable values to the CRT. The channel is declared of type *broadcast*, which is weakly synchronized and therefore does not require a receiver automaton [4]. In order to be visible on the Spread network, the counter variables and the counter channel follow the naming conventions of DTRON as explained earlier. The counter channel has to be synchronized at some point for the CRT to be able to produce a coverage report. To achieve this, the tool adds a process to the system, containing one location with a self-edge, that synchronizes the counter channel periodically.

The second functionality of the CRT is to connect to the Spread network and monitor the counter variables and to build statistics about the edges visited during the test run. At the moment, CRT provides support for *edge*, *edge-pair* and *requirement coverage*, respectively. However, other structural coverage criteria could be implemented in the tool. The requirement coverage criterion, is a simplified version of the edge coverage one described above, in which test targets fulfilling certain system requirements are manually added to the model as counter variables.

If we would follow an offline test generation approach, a set of traces satisfying, e.g. edge coverage, can be easily obtained via model-checking of the property $E <> i\_c1 \&\& \ldots \&\& i\_c20$ using tools like *verifyta* provided by UPPAAL as described in [14].

Code coverage. Since we have access to the source code of the IUT we also track how much of the code has been covered by each test run. For this purpose, we use the *coverage* tool for Python [15]. The tool counts the number of statements in the source files and monitors which of them are executed. At the end of the test run, it provides a coverage report detailing the coverage level for different source files.

V. EVALUATION

MBT has two distinct components: modeling and test generation. Each of them brings its own benefits to reducing the effort of the testing process. The main benefit of modeling is that it forces the designers to simulate and verify the system specification before deciding to implement a new feature. During the specification and development of the MBPeT tool, we detected many specification inconsistencies which could have resulted otherwise time spent during the implementation, testing and debugging.

The automatic test design has also its benefits, even if the test suite has to be generated for each iteration. Due to the online generation approach and also of the available tool support, generating only parts of the test suite when the model changes has not been considered. We do not consider it a problem for two reasons: 1). with our approach the duration of a test run requires less than one minute to achieve an acceptable edge coverage level and 2) if the models would become too complex to handle timely test generation, then raising the level of abstraction or focusing the test generation on certain parts of the models will help.

However, if complex test suites cannot be avoided there exist a body of work which has addressed regression testing in the context of MBT, e.g., [16], [17], [18], in its vast majority targeted to offline test generation. In addition, there exist already commercial tools such as Conformiq tool-set [19], which optimizes the offline test suite generation by generating only new test cases and removing old test cases which are not relevant anymore.

With respect to our case study, the code base of the Master node is approx. 2100 LOC written in Python, whereas the test adapter needed for the models in Figures 2 and 3 is slightly over 200 LOC written in Java.

Letting the DTRON tool randomly generate tests from the model in Figure 2, we could identify six different test scenarios, as depicted in Table I. For each scenario, we extracted the corresponding edge coverage and statement coverage levels. As shown in this table, the minimal edge coverage achieved for our particular models is 70%, when running with one slave which does not saturate. This corresponds to 91% statement coverage. The highest edge coverage, 95%, is achieved when having more than two slaves, of which at least one saturates, one generates load unsaturated, and one is idling. In this case, the statement coverage increased to 98% coverage.

With the model in Figure 2, full edge coverage cannot be achieved due to two mutually exclusive paths in the model: for a given set of slaves one cannot have both an idling slave (which would be killed via edges `c18`, `c19`) and all slaves saturated (edge `c14`).

Test Scenario	Covered Edges	EC	Statement Coverage
1 slave, no saturation	1-11, 15, 17, 20	70%	91%
1 slave, saturation	1-11, 14, 15, 17, 20	75%	91%
2 slaves, slave 1 saturated	1-13, 15, 16, 17, 20	85%	92%
2 slaves, both saturated	1-17, 20	90%	91%
2 slaves, 1 idle	1-13, 15, 17-20	90%	92%
>2 slaves, 1 idle	1-13, 15-20	95%	98%

TABLE I
COVERAGE RESULTS

Due to the way the tests are generated from these models, with each test run we may obtain a different trace depending on the number and behavior of the slaves. However, using two slave nodes in the test configuration, provided an acceptable edge coverage level. Adding another slave will increase the edge coverage by 5% and the statement coverage by 6%. At the moment, we did not consider this approach necessary, since when inspecting the source code coverage for all three test scenarios using two slave nodes, we found that actually the entire code base was covered by the respective traces.

VI. Conclusion and Future Work

We have presented an approach in which model-based testing and continuous integration approaches have been combined in order to lessen the testing effort and consequently shorten the integration cycles.

Having performed simulation and verification of the specifications increased their quality and decreased the number of failures originating in the specifications, such as common mistakes, omissions and misinterpretations of the requirements. The UPTA formalism allowed us the modeling of time and the verification of time properties. Using automatically generated tests decreased the time spent to develop tests every time a new update was performed either in the specification or in the implementation.

Since the repositories can be updated independently, the modeling and development teams are immediately aware of problems in the build. Ideally, the implementation and the models should be in sync, that is the implementation should reflect the model. As long as the tests conclude that the implementation and models conform to each other the builds are successful. If they start diverging, we can conclude that either the model or implementation is erroneous, or the other team has not yet updated their part of the system to conform to new requirements.

In our current case study, we used tests generated and executed on-the-fly. This approach has both advantages and disadvantages. As explained in the paper, one benefit is that using online MBT allows for non-deterministic behavior due for instance to concurrency or to time/value domains. In addition, it does not require an additional test execution framework to be included in the toolchain, although it does require the implementation of the adapter. However, the adapter has to be updated only when new observable interfaces are added to the SUT, otherwise it can be reused as such.

Among the perceived drawbacks of online MBT is that tests have to be regenerated from scratch every time, which can be time consuming. However, since all the tests are generated automatically, the generation times are short for reasonably sized models. If the models become too complex, increasing the level of abstraction or focusing only on certain parts of the functionality should be considered. For instance, with our models, the average test run is on average less than a minute. This means that one can get a test report in several minutes since a new version is committed to the SCM servers. Another drawback of our online MBT approach is that the test session is stopped on the first failure of the tests, which compared to offline testing will not give a good overview of the failed/passed test case ratio. However, observing the achieved coverage with the CRT tool alleviated this problem and allowed us to identify which parts of the specification passed testing and which did not.

Future work will look into more detail at using offline MBT, and, in particular in deploying more efficient methods of model and test suite update via the modularization of test specifications. Also, by improving test reporting, more meaningful debug information can be provided for the development teams.

References

[1] M. Fowler. (2006, May) Continuous integration. Retrieved: 20.08.2013. [Online]. Available: http://martinfowler.com/articles/continuousIntegration.html

[2] P. M. Duvall, S. Matyas, and A. Glover, *Continuous integration: improving software quality and reducing risk*. Addison-Wesley Professional, 2007.

[3] R. Rogers, "Scaling continuous integration," in *Extreme Programming and Agile Processes in Software Engineering*, ser. Lecture Notes in Computer Science, J. Eckstein and H. Baumeister, Eds. Springer Berlin Heidelberg, 2004, vol. 3092, pp. 68–76. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-24853-8_8

[4] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *Formal Methods for the Design of Real-Time Systems: 4th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT 2004*, ser. LNCS, M. Bernardo and F. Corradini, Eds., no. 3185. Springer–Verlag, September 2004, pp. 200–236.

[5] M. Utting and B. Legeard, *Practical Model-Based Testing – A Tools A pproach*. San Francisco, CA, USA: Morgan Kaufmann, 2007.

[6] ITEA 2, "D-MINT project result leaflet: Model-based testing cuts development costs," http://www.itea2.org/project/result/download/result5519?file=06014_D_MINT_Project_Leaflet_results_oct_10.pdf, February 2010, retrieved: 20.08.2013.

[7] G. J. Myers *et al.*, *The Art of Software Testing*. John Wiley & Sons, Hoboken, NJ, 2nd ed edition, 2004.

[8] T. Ahmad, F. Abbors, D. Truscan, and I. Porres, "Model-based performance testing using the MBPeT Tool," Turku Centre for Computer Science, TUCS Technical Reports 1066, 2013.

[9] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking for real-time systems," in *Logic in Computer Science, 1990. LICS'90, Proceedings., Fifth Annual IEEE Symposium on Logic in Computer Science*. IEEE, 1990, pp. 414–425.

[10] Jenkins CI - Meet Jenkins. Online at https://wiki.jenkins-ci.org/display/JENKINS/Meet+Jenkins. Retrieved: 20.08.2013.

[11] Subversion. Online at http://subversion.apache.org/. Retrieved: 20.08.2013.

[12] A. Anier and J. Vain, "Model based continual planning and control framework for assistive robots." in *PECCS 2012 - Proceedings of the 2nd International Conference on Pervasive Embedded Computing and Communication Systems*, C. Benavente-Peces, F. H. Ali, and J. Filipe, Eds. SciTePress, 2012, pp. 403–406. [Online]. Available: http://dblp.uni-trier.de/db/conf/peccs/peccs2012.html

[13] The Spread Toolkit - Overview. Online at http://spread.org/SpreadOverview.html. Retrieved: 20.08.2013.

[14] A. Hessel, K. G. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou, "Testing real-time systems using UPPAAL," in *Formal methods and testing*, R. M. Hierons, J. P. Bowen, and M. Harman, Eds. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 77–117. [Online]. Available: http://dl.acm.org/citation.cfm?id=1806209.1806212

[15] Code coverage measurement for Python – coverage, v. 3.6. Online at https://pypi.python.org/pypi/coverage. Retrieved: 20.08.2013.

[16] B. Jiang, T. Tse, W. Grieskamp, N. Kicillof, Y. Cao, and X. Li, "Regression testing process improvement for specification evolution of real-world protocol software," in *Quality Software (QSIC), 2010 10th International Conference on*. IEEE, 2010, pp. 62–71.

[17] E. Fourneret, F. Bouquet, F. Dadeau, and S. Debricon, "Selective test generation method for evolving critical systems," in *Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on*. IEEE, 2011, pp. 125–134.

[18] Y. Chen, R. L. Probert, and H. Ural, "Model-based regression test suite generation using dependence analysis," in *Proceedings of the 3rd international workshop on Advances in model-based testing*, ser. A-MOST '07. New York, NY, USA: ACM, 2007, pp. 54–62. [Online]. Available: http://doi.acm.org/10.1145/1291535.1291541

[19] Conformiq tool set. Online at http://www.conformiq.com/. Retrieved: 20.08.2013.

Static Analysis Techniques and Tools: A Systematic Mapping Study

Vinícius Rafael Lobo de Mendonça
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: viniciusmendonca@inf.ufg.br

Cássio Leonardo Rodrigues
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: cassio@inf.ufg.br

Fabrízzio Alphonsus A. de M. N. Soares
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: fabrizzio@inf.ufg.br

Auri Marcelo Rizzo Vincenzi
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: auri@inf.ufg.br

***Abstract*—The main disadvantage of static analysis tools is their high false positive rates. False positives are errors that either do not exist or do not lead to serious software failures. Thus, the benefits of automated static analysis tools are reduced due to the need for manual interventions to assess true and false positive warnings. This paper presents a systematic mapping study to identify current state-of-the-art static analysis techniques and tools as well as the main approaches that have been developed to mitigate false positives.**

***Keywords*-automatic static analysis; false positive; systematic mapping study.**

I. Introduction

There are two Verification and Validation (V&V) approaches: dynamic and static [1]. The first approach is characterized by software implementation and defect detection through assessment of program outputs, which is similar to software testing. In the second approach, program execution is not required, and identification of potential faults is carried out through evaluation of software source codes, design diagrams, requirements, etc. Inspections and code reviews are types of techniques used in static analysis but, in general, they are performed by human. The focus of this work is the evaluation of techniques and tools used to perform automated static analysis.

Automated static analysis vocabulary includes the following terms: false positives, true positives and false negatives. A false positive occurs when a tool alerts to the presence of a non-existent fault. A false negative occurs when a fault exists, but it is not detected due to the fact that static analysis tools are not perfectly accurate and may not detect all errors. Finally, a true positive occurs when a tool produces a warning to indicate the presence of a real defect in the product under analysis.

Examples of automated static analysis tools are FindBugs [2], PMD [3] and CheckStyle [4]. The disadvantage of these tools is that they produce a high rate of false positives, whereas developers are only interested in true errors, which are the ones that require correction. False positive alerts lead to an increase in process costs, because their detection is usually done by human intervention. This consumes precious time that could be used for the correction of real faults [5]. Regardless of such disadvantage, static analysis tools are very useful for carrying out initial verification and validation activities compared to other quality assurance procedures, especially due to their low implementation cost.

We conducted a Systematic Mapping Study (SMS) of static analysis techniques and tools to investigate how they avoid false positives. A comprehensive data extraction process and classification of the primary studies provided answers to our research questions. The remaining of this paper is organized as follows: Section II describes the methodology used to conduct the systematic mapping and shows the results obtained in each phase. Section III reveals main results and answers to the research questions defined in Section II. Finally, Section IV shows our conclusions and implications for future studies.

II. Background

This paper shows the development of a systematic map based on the process presented by Petersen [6]. It is composed of the following steps: i) definition of research questions, ii) analysis of relevant studies, iii) study selection, iv) keywording of abstracts, v) data extraction, and vi) mapping process (Figure 1). To increase the reliability of our proposed SMS, some of the guidelines provided by Kitchenham et al. [7] were followed, such as the use of control studies to assess search string quality and Quality Assessment Strategy [8].

A. Research Questions

Our systematic mapping identifies relevant papers on static analysis. We aim to understand the behaviour of automated static analysis tools as well as find out which of the proposed methods mitigate false positives, if any.

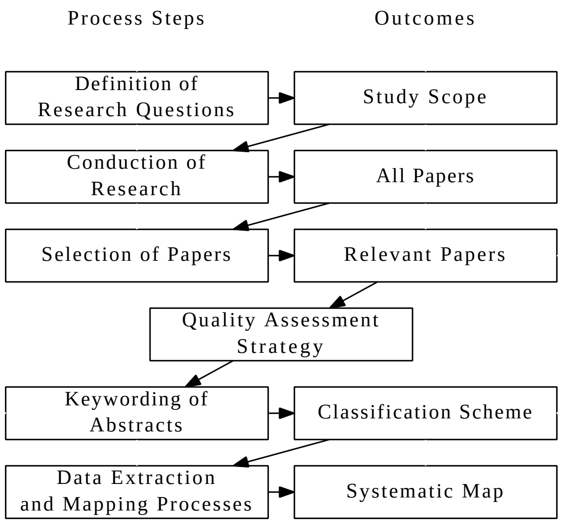


Figure 1. Steps of a Systematic Mapping Study - Adapted from [6].

Each question is answered according to the following criteria: Population, Intervention, Control and Results. The criteria comparison is not applicable to this research, so it was not used. Information on each of them is found in [7]. Our research questions are:

- RQ$_1$: Which static analysis tools and approaches are used to reduce false positives?
- RQ$_2$: Which types of warnings are emitted by the tools and which static analysis methods are employed?
- RQ$_3$: Should various static analysis techniques be used in combination to reduce false positives?

B. Search Strategy, Data Sources and Study Selection

The search strategy involved the creation of a string to operate upon scientific digital libraries for the selection of primary studies. The digital libraries selected were: ACM, IEEE, Engineering Village (Compendex) and SpringerLink. The search string presented in Figure 2 was applied to search engines by using the advanced search mechanism and, in some cases, it was tailored for a specific search engine. The papers used as controls are listed in Table I, it was developed from generic terms found in the control articles used. Static analysis does not have a broad vocabulary, similar terms are usually found in papers, resulting in small search string. The general search string is as follows:

```
((static analysis OR bug finding OR static code analysis OR find bug
OR analysis static) AND (false positive OR warnings))
```

Figure 2. The general search string.

After excluding control articles and their repeated retrievals, the string used in databases returned 615 primary studies, all of them catalogued in a specialized tool, named State of the Art through Systematic Review (StArt) [9]. The results extracted from StArt revealed that ACM retrieved the highest rate of primary studies (52%), followed by Engineering Village (23%), IEEE (17%) and SpringerLink (8%). The main data retrieved from each primary study was stored using the JabRef tool [10] for further classification.

TABLE I. CONTROL PAPERS

#	Title	Citation	Consultation Database
CA$_1$	*Which Warnings Should I Fix First?*	[11]	ACM
CA$_2$	*Finding Bugs is Easy*	[5]	ACM
CA$_3$	*Comparing Bug Finding Tools with Reviews and Tests*	[12]	SpringerLink
CA$_4$	*A Comparison of Bug Finding Tools for Java*	[13]	IEEE
CA$_5$	*Static Code Analysis*	[14]	IEEE

The inclusion criteria for this study are:

- IC$_1$: Primary studies analysing the warnings emitted by static analysis tools;
- IC$_2$: Primary studies proposing methods/tools to reduce false positives;
- IC$_3$: Primary studies comparing static analysis tools/methods;

The exclusion criteria are:

- EC$_1$: Primary studies on static analysis that do not assess warnings or reduce false positives;
- EC$_2$: Primary studies that are repeated retrievals or contain a maximum of two pages;
- EC$_3$: Primary studies that cannot be accessed;
- EC$_4$: Primary studies that do not use static analysis;
- EC$_5$: Primary studies that are not written in English or Portuguese.

The primary studies underwent two other stages of selection. In the second stage, EC$_2$, EC$_3$ and EC$_5$ were applied. This action ensured a significant reduction in the number of studies, after which 270 studies remained. In the third and last stage, each study's title and abstract were assessed by applying EC$_1$ and EC$_4$, eliminated respectively 40% and 33% of the primary studies, remaining 64 papers for quality assessment strategy (Figure 3).

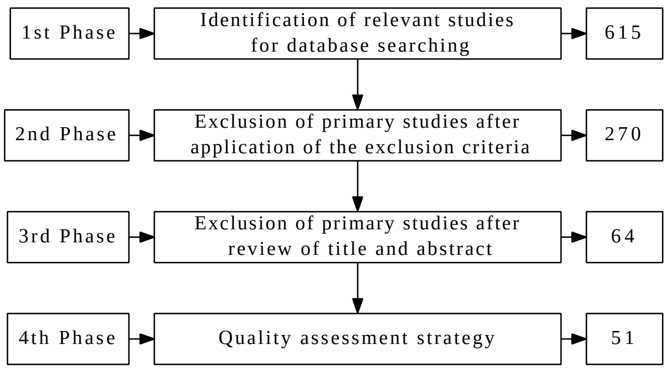

1st Phase
Identification of relevant studies for database searching
615
2nd Phase
Exclusion of primary studies after application of the exclusion criteria
270
3rd Phase
Exclusion of primary studies after review of title and abstract
64
4th Phase
Quality assessment strategy
51

Figure 3. Four stages for selection of primary studies

C. Quality Assessment Strategy and Classification of Selected Studies

After application of the previous steps, 64 papers were selected. Aiming at increasing and ensuring the quality of the developed work, we decided to assess the quality of the selected primary studies based on the criteria created by Dybå and Dingsøyr [8]. These criteria are composed of eleven question but we use only eight since three of them are out of the scope of our work (questions 5, 6 and 9). Information on each of them is found in [8]. It was assigned 1 when the primary study satisfies the criterion and 0 otherwise. Observe that criteria 1 and 2 are considered exclusion criteria since a primary study not satisfying them implies it should be dropped of. Table II presents the final results of the quality assessment. Among 51 articles, 9 (17%) received the maximum score (8) indicating that only a few number of primary studies are concerned to document the stages of the development of their research. 90% of the primary studies collected data to answer the research question, but only 40% of the work are worried to validate the collected data, and 41% are worried to show the results clearly. Observe that the lack of such information makes difficult to draw conclusions about these research areas. Primary studies were classified into three classes, identified after reading the 51 remaining papers: research type, approach type, and types of false positive errors.

In the first class, works were classified based on the types of research: validation research, evaluation research, solution proposal, philosophical papers, opinion papers and experience papers. Information on each of them is found in [6]. The second class was defined to answer RQ_1. Therefore, a classification scheme was developed based on the terminology used in the primary studies. The types of approaches identified were: comparative study, algorithm, new tool, improve existing tool, ranking error, hybrid approach, technique and method. A main approach type was defined for each primary study, which means that it may or may not contain features of other less relevant types of approaches. The characteristics of each approach type are:

- Comparative Study (CS): A primary study that compares static analysis approaches to identify cases in which application of an approach is better than another;
- Algorithm (A): A primary study that proposes a new algorithm that may or may not be used in combination with a static analysis tool to reduce false positives;
- New Tool (TL): A primary study that proposes a new tool which may be more effective than existing tools or used in combination with other tools to identify false positives not yet detected;
- Improve Existing Tool (IT): A primary study that proposes an improvement of an existing tool. For instance, a new bug pattern;
- Ranking Error (RE): A primary study that proposes a new ranking or an improved technique to rank error reports;
- Hybrid Approach (HA): A primary study that combines static and dynamic approaches to reduce the disadvantages of each technique aiming at false positive mitigation;
- Technique (T): A primary study that aims to find a solution to a specific problem [15];
- Method (M): A primary study that searches for a general solution to a problem [15].

The third class is related to false positive errors. It was developed to identify the primary focus of our study. Errors were classified as interface fault, data fault, cosmetic fault, initialization fault, control fault, and computation fault. This classification was created by Basili and Selby [16], where one can find more information on each class of fault. However, most of the 64 selected studies mention the false positives generated by static analysis tools in a general way. Thus, a new error category was created: extensive study. The primary studies that identify more than 10 error types were also included in this category. Studies containing 2-9 defects were classified into several categories. Also, when more than one defect of the same type was detected, only one would be considered. Most primary studies aimed at reducing false positives of various types. This is a relevant result, because the purpose of our systematic mapping is to provide an overview of the research field.

D. Data Extraction and Mapping Processes

The JabRef and spreadsheets were used in the remaining 51 primary studies to create a three-class classification: research type, error type, and approach type. Figure 4 shows that 25 out of 51 (49%) primary studies were classified as solution proposals. This indicates that many proposed new approaches or improvements for existing ones aimed at reducing false positive alerts emitted by static analysis tools. However, none of them was classified as validation research, indicating that there is no experimental validation of the proposed approaches.

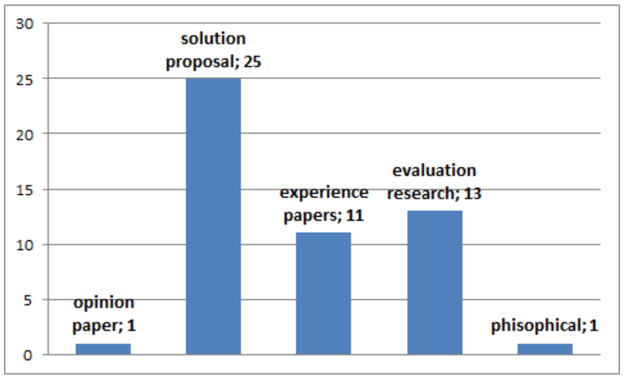


Figure 4. The classification of selected papers in relation research classes

Regarding the types of approaches identified, there were a variety of proposed solutions to problems, mainly methods (18.91%) and new tools (16.21%), which are shown in Figure 5). This indicates that many researchers, who were not satisfied with existing tools, proposed new ones as well as prototypes to assist developers. Some works also presented

TABLE II. RESULTS OF SYSTEMATIC MAPPING STUDY QUALITY EVALUATION AND THE AMOUNT OF THE FALSE POSITIVES ACHIEVED IN EACH PRIMARY STUDY

Paper	RE	AI	CO	RD	DC	DA	FI	VA	RS	TFP	A	FPR(%)
[17]	1	1	1	1	1	0	1	0	6	S	T	63%
[18]	1	1	1	1	1	0	0	1	6	ND	T	0%
[19]	1	1	1	1	1	1	1	1	8	S	T	20%
[20]	1	1	1	1	1	0	0	1	6	S	T	BCI 13%
[21]	1	1	1	1	1	0	0	1	6	S	T	IINC
[22]	1	1	1	1	1	1	1	1	8	S	RE	19%
[23]	1	1	1	1	1	1	1	1	8	S	RE	BCI 0%
[24]	1	1	0	1	1	0	0	1	5	S	RE	IINC
[25]	1	1	1	1	1	0	1	1	7	S	M	IINC
[26]	1	1	0	1	1	0	1	1	6	SB	M	0%
[27]	1	1	0	1	1	1	1	1	7	S	M	IINC
[28]	1	1	1	1	1	1	0	1	7	S	M	IINC
[29]	1	1	1	1	1	1	1	0	7	S	M	32%
[30]	1	1	1	1	1	1	1	1	8	S	M	BCI 74%
[31]	1	1	0	1	1	0	1	1	6	S	M	IINC
[32]	1	1	0	1	1	1	1	0	6	S	M	BCI 0%
[33]	1	1	1	1	1	1	0	1	7	S	M	IINC
[34]	1	1	0	1	0	0	0	0	3	S	M	34%
[35]	1	1	0	1	1	1	1	1	7	C	M	IINC
[36]	1	1	1	1	0	0	1	0	5	S	IT	15%
[37]	1	1	1	1	1	1	1	1	8	S	IT	IINC
[38]	1	1	1	1	0	0	0	1	5	S	IT	BCI 0%
[39]	1	1	0	1	1	0	0	1	5	S	IT	IINC
[40]	1	1	1	1	1	1	1	1	8	S	IT	IINC
[41]	1	1	0	1	1	1	1	1	7	S	TL	BCI 0%
[42]	1	1	0	1	1	0	0	0	4	S	TL	IINC
[43]	1	1	1	1	1	0	0	0	5	IB	TL	IINC
[44]	1	1	1	1	1	1	1	1	8	BO	TL	IINC
[45]	1	1	1	0	1	0	0	1	5	S	TL	IINC
[46]	1	1	0	1	1	0	0	0	4	S	TL	IINC
[47]	1	1	1	1	1	1	1	1	8	C	TL	IINC
[48]	1	1	1	1	1	0	0	0	5	D	TL	IINC
[49]	1	1	0	1	1	0	0	0	4	S	TL	IINC
[50]	1	1	0	1	1	1	0	0	5	S	TL	IINC
[51]	1	1	0	1	1	0	0	0	4	CS		
[52]	1	1	0	1	1	0	0	0	4	CS		
[53]	1	1	0	1	1	0	0	1	5	CS		
[54]	1	1	0	1	0	0	0	0	3	CS		
[55]	1	1	1	1	1	0	0	0	5	CS		
[56]	1	1	0	1	1	0	1	0	5	CS		
[57]	1	1	1	0	1	0	0	1	5	DR	A	BCI 0 %
[58]	1	1	0	1	1	0	0	1	5	DR	A	IINC
[59]	1	1	1	1	1	0	0	1	6	S	A	IINC
[60]	1	1	0	1	1	0	0	0	4	S	A	BCI 20%
[61]	1	1	0	1	1	1	0	1	6	S	A	IINC
[62]	1	1	1	1	1	0	0	0	5	S	A	BCI 38%
[63]	1	1	0	0	1	0	0	1	4	S	HA	IINC
[64]	1	1	0	1	1	1	0	0	6	S	HA	IINC
[65]	1	1	1	1	1	1	0	0	6	S	HA	IINC
[66]	1	1	1	1	1	1	0	1	7	S	HA	IINC
[67]	1	1	0	1	1	0	0	0	4	S	HA	IINC
RS	51	51	28	49	46	20	21	32	*	*	*	*

Acronyms		
RE: RESEARCH	VA: VALUE	SB: STRING BUG
AI: AIM	RS: RESULT	C: CONCURRENCE
CO: CONTEXT	TFP: TYPE OF FALSE POSITIVE	IB: INTEGER BUG
RD: RESEARCH DESIGN	A: APPROACH	BO: BUFFER OVERFLOW
DC: DATA COLLECTION	FPR: FALSE POSITIVE RATE	RC: DATA RACE
DA: DATA ANALYSIS	S: SEVERAL	BCI: BEST CASE IS
FI: FINDINGS	ND: NULL DEFERENCE	IINC: IT IS NOT CLEAR

improvements for the FindBugs tool [2], which is used for static analysis of Java programs. Most primary studies suggest ways to mitigate false positives of various types (Figure 5). The fact that a large number of research proposals focus on the many kinds of errors exposed by static analysis contributes to the provision of a variety of mitigation techniques.

A bubble chart was designed to display the intersection of two classes: approach type and types of false positives (Figure 5). According to Petersen et al. [6], the bubble chart was effective in the sense that it gave an overview of the research field and produced a map of results.

III. MAIN FINDINGS

In this section, we present the answers to the three proposed research questions based on the primary studies found.

A. Answer to RQ$_1$ – Tools and Approaches

With respect the tools, it can be observed that a significant number of static analysis tools is used by researchers trying to reduce false positives, i.e., there is not a consensus that the tool A is better or worse than the tool B, or whether it is better to use an open source tool or a commercial tool. Some authors believe that the use of tools is not mutually exclusive because if tool A finds more real faults of "null deference" than a tool B, but B is better than A on detecting real faults of "buffer overflow", then the correct choice should be to combine both tools taking the advantage of both. This strategy aims to potentiating the strengths of each tool, increasing the amount of real defects found, and also the precision, because if more than one tool reports a same warning on the same line, this may indicates that the probability of the warning corresponds to a true positive is greater than the one reported by a single tool.

The collected data indicate that 16.21% of the papers propose a new static analysis tool, the majority without any comparison with other existing tool to demonstrate the effectiveness of the proposed approach. From this percentage, just 25% seek to mitigate defects in a comprehensive way, the other 75% remaining seek to reduce false positive of a specific class. This large concentration of new tools with a focus on some specific defect demonstrates a possible deficiency of existing tools for detecting such defects. The main defects found are "data fault" and "initialization fault" representing together 50% of the works proposed of improved tool.

There are also 12.16% of the studies which present hybrid techniques by combining dynamic and static analysis. According to Aggarwal and Jalote [45], IT community believes that the static and dynamic approaches are complementary. Static analyzers are faster and simpler to use than dynamic ones. Moreover, they also help to identify problems earlier in the development process, when the cost to correct them is lower but, in general, static analyzers generate large amounts of false positives and false negatives. On the other hand, dynamic analyzers are accurate and generate few false positives, but to test all possible conditions in a program with thousands of line of code is practically impossible. During the systematic mapping were found works which use both approaches in a complementary way to reduce the drawbacks of each other.

Among the tools used or cited in the primary studies, those that stand out are: FindBugs and Checkstyle. Both are used together with PMD by the quality platform SonarQube for the calculation of part of its static metrics, but SonarQube does not seek to reduce false positives.

B. Answer to RQ$_2$ – The types of false positive errors

We also tried to identify specific classes of false positives warnings but the majority of the works (33.78%) fell down in the generic classification of false positives, i.e., the information is not available. The fact that significant amount of work develops techniques or tools to mitigate various false positives is something negative demonstrating the existence of a large gap in this research area to be filled. 20% of the proposed methods provided generic methods which are palliative solutions, not solving the problem of the high rate of false positive efficiently.

Table II shows a summary of the type of false positive, the technique and the false positive rate of the 51 primary studies investigated. Excluding the comparative studies that do

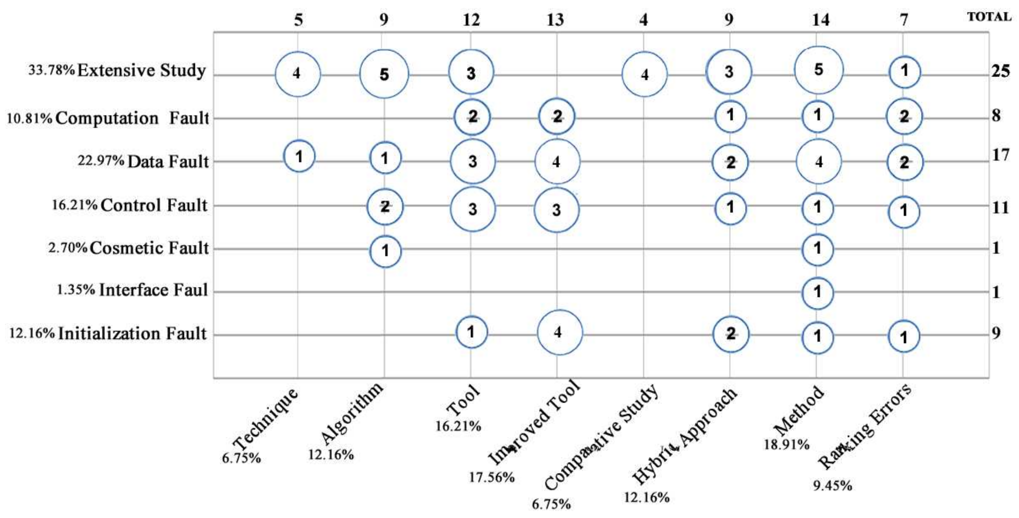


Figure 5. Bubble chart – Types of Approaches × Types of False Positives

not investigate the effectiveness of a specific approach, 63% of the remaining works do not make clear how the proposed approach was efficient, they just mention that the false positive rate was reduced without any evidences. But, how much is the reduction? Without this information, the proposed approach has no change to be addopted on real software development environments. Therefore, improve the way the experiments are conducted on these research area is of fundamental importance to provide such an evidences.

Maybe the real solution does not treats in reduce the rate of false positives, but reducing the amount of certain types of false positives. Treating this problem broadly, your solution may be far from being achieved. One approach might be to assign a weight or priority for each type of false positive, characteristic established in some bug-finding tools (FindBugs, PMD, and Checkstyle are some of them).

In this manner, bugs with greater weight, hence higher priority should be given greater attention and other bugs with lower priority can be observed or subsequently depending on how much low is your priority should be ignored. Currently, we live in a scenario that programs are getting ever larger, exceeding millions of line of code, so the task of analyzing the warnings emitted by bug finding tools, should be performed in a smarter way, reducing the human effort and time on this activity. To do this, we think an important step is to valorize the types of false positives but, among existing tools, there is no consensus between the weight or priority, and similar or identical bugs may have different values in different tools.

C. Answer to RQ_3 – Application of Static Analysis Tools

The answer to reduce false positives rate might be resumed in one word: combine. It is not necessary to create something new or improve something that already exists, but the strategy should be to combine static analysis tools and/or dynamic and static tools and to conduct significant empirical studies to identify the best combination of tools to reduce drawbacks and to increase benefits. Unfortunately, the task is not so easy since there are several technical details to be overcome to create a meta static analysis tool, such as, how to combine different warning's prioritization classification in a single meta tool; how to manage different output formats; and so on.

Rutar et al. [13] is an example of primary study which uses this approach. They utilizes tools in conjunction with applications of different sizes. Each tool performs a different balancing to equilibrate the real location of errors, the generation of false positives and true positives and, consequently, there is little overlap among the generated warnings. These different approaches of balancing can involve the necessity of using multiple tools in the verification of an application. Then, Rutar et al. [13] suggested the development of a meta-tool, which combines the results of different tools for searching errors.

IV. Final Considerations

This paper showed the development of a systematic mapping study on static analysis approaches and tools aiming at reducing the number of false positives generated. After the selection of 51 studies, the mapping combined protocol processes developed by Petersen et al. [6] and Kitchenham et al. [7]. The selected works provided a variety of static analysis approaches, including proposals for improving existing tools. FindBugs stands out in the sense that many primary studies not only use it, but also discuss how it may be improved.

Among the retrieved studies, there was a lack of works on the types of false positive errors and the tools that generate them. This kind of research would help developers identify the tools that best serve their needs. The mapping also revealed studies that use hybrid approaches, which combine static and dynamic analyses techniques. Furthermore, a combination of different static analysis approaches proved more efficient than their isolated use.

Based on the mapping results and due to a lack of validation research on the subject, we propose a large-scale experimental study aiming at finding answers to open questions. This would contribute to advancements in the use of static analysis tools in the early stages of the development cycle, as well as identification of the types of defects that should be treated by other verification and validation techniques. Moreover, the development of a methodology for combining static analysis approaches and tools is also recommended for future research.

ACKNOWLEDGMENT

The authors would like to thank the Instituto de Informática – INF/UFG, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – CAPES – Brasil, and Fundação de Amparo à Pesquisa do Estado de Goiás – FAPEG – Brasil, which support this work.

REFERENCES

[1] I. Sommerville, S. Melnikoff, R. Arakaki, and E. de Andrade Barbosa, *Software Engineering*. ADDISON WESLEY BRA, 2008, [retrieved: Oct., 2013]. [Online]. Available: http://books.google.com.br/books?id=ifIYOgAACAAJ

[2] FindBugs, [retrieved: Oct., 2013]. [Online]. Available: http://findbugs.sourceforge.net/

[3] PMD, [retrieved: Oct., 2013]. [Online]. Available: pmd.sourceforge.net/

[4] CheckStyle, [retrieved: Oct., 2013]. [Online]. Available: http://checkstyle.sourceforge.net

[5] D. Hovemeyer and W. Pugh, "Finding bugs is easy," *SIGPLAN Not.*, vol. 39, no. 12, pp. 92–106, 2004.

[6] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *12th International Conference on Evaluation and Assessment in Software Engineering*, vol. 17, 2008, p. 1.

[7] B. Kitchenham and S. Charters, "Guidelines for performing Systematic Literature Reviews in Software Engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.

[8] T. D. Aĕ and T. D. Aÿyr, "Empirical studies of agile software development: A systematic review," *Information and Software Technology*, vol. 50, pp. 833–859, 2008.

[9] StArt, [retrieved: Oct., 2013]. [Online]. Available: http://lapes.dc.ufscar.br/tools/start-tool

[10] JabRef, [retrieved: Oct., 2013]. [Online]. Available: http://jabref.sourceforge.net/

[11] S. Kim and M. D. Ernst, "Which warnings should i fix first?" in *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, ser. ESEC-FSE'07. New York, NY, USA: ACM, 2007, pp. 45–54.

[12] S. Wagner, J. Jürjens, C. Koller, and P. Trischberger, "Comparing bug finding tools with reviews and tests," in *Proceedings of the 17th IFIP TC6/WG 6.1 international conference on Testing of Communicating Systems*, ser. TestCom'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 40–55.

[13] N. Rutar, C. B. Almazan, and J. S. Foster, "A comparison of bug finding tools for java," in *Proceedings of the 15th International Symposium on Software Reliability Engineering*, ser. ISSRE'04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 245–256.

[14] P. Louridas, "Static code analysis," *Software, IEEE*, vol. 23, no. 4, pp. 58–61, 2006.

[15] L. Nascimento, D. Viana, P. Silveira Neto, D. Martins, V. Garcia, and S. Meira, "A systematic mapping study on domain-specific languages," in *ICSEA'12, The Seventh International Conference on Software Engineering Advances*, 2012, pp. 179–187.

[16] V. Basili and R. Selby, "Comparing the effectiveness of software testing strategies," *Software Engineering, IEEE Transactions on*, vol. SE-13, no. 12, pp. 1278–1296, 1987.

[17] M. G. Nanda and S. Sinha, "Accurate interprocedural null-dereference analysis for java," in *Proceedings of the 31st International Conference on Software Engineering*, ser. ICSE'09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 133–143.

[18] P. Emanuelsson and U. Nilsson, "A comparative study of industrial static analysis tools," *Electron. Notes Theor. Comput. Sci.*, vol. 217, pp. 5–21, Jul. 2008.

[19] Lucia, D. Lo, L. Jiang, and A. Budi, "Active refinement of clone anomaly reports," in *Proceedings of the 2012 International Conference on Software Engineering*, ser. ICSE'12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 397–407.

[20] S. S. Heckman, "Adaptively ranking alerts generated from automated static analysis," *Crossroads*, vol. 14, no. 1, pp. 7:1–7:11, Dec. 2007.

[21] S. Heckman and L. Williams, "A model building process for identifying actionable static analysis alerts," in *Proceedings of the 2009 International Conference on Software Testing Verification and Validation*, ser. ICST'09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 161–170.

[22] A. Vetro, M. Morisio, and M. Torchiano, "An empirical validation of findbugs issues related to defects," Durham University – Grey College, Durham, Apr., pp. 144–153.

[23] E. Bodden and K. Havelund, "Aspect-oriented race detection in java," *IEEE Trans. Softw. Eng.*, vol. 36, no. 4, pp. 509–527, Jul. 2010.

[24] G. Liang, L. Wu, Q. Wu, Q. Wang, T. Xie, and H. Mei, "Automatic construction of an effective training set for prioritizing static analysis warnings," in *Proceedings of the IEEE/ACM international conference on Automated software engineering*, ser. ASE'10. New York, NY, USA: ACM, 2010, pp. 93–102.

[25] S. Kim, T. Zimmermann, K. Pan, and E. J. J. Whitehead, "Automatic identification of bug-introducing changes," in *Proceedings of the 21st IEEE/ACM International Conference on Automated Software Engineering*, ser. ASE'06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 81–90.

[26] P. Chen, Y. Wang, Z. Xin, B. Mao, and L. Xie, "Brick: A binary tool for run-time detecting and locating integer-based vulnerability," in *Availability, Reliability and Security, 2009. ARES'09. International Conference on*, 2009, pp. 208–215.

[27] D. Babi? and A. J. Hu, "Calysto: Scalable and precise extended static checking," 2008.

[28] C. Csallner and Y. Smaragdakis, "Check 'n' crash: combining static checking and testing," in *Proceedings of the 27th international conference on Software engineering*, ser. ICSE'05. New York, NY, USA: ACM, 2005, pp. 422–431.

[29] A. Fehnker, R. Huuck, and S. Seefried, "Concurrency, compositionality, and correctness," D. Dams, U. Hannemann, and M. Steffen, Eds. Springer-Verlag, 2010, ch. Counterexample guided path reduction for static program analysis, pp. 322–341.

[30] F. Ivancic, G. Balakrishnan, A. Gupta, S. Sankaranarayanan, N. Maeda, H. Tokuoka, T. Imoto, and Y. Miyazaki, "Dc2: A framework for scalable, scope-bounded software verification," in *Automated Software Engineering (ASE), 2011 26th IEEE/ACM International Conference on*, 2011, pp. 133–142.

[31] S. Kim, H. Zhang, R. Wu, and L. Gong, "Dealing with noise in defect prediction," in *Proceedings of the 33rd International Conference on Software Engineering*, ser. ICSE '11. New York, NY, USA: ACM, 2011, pp. 481–490.

[32] S. Lu, S. Park, and Y. Zhou, "Detecting concurrency bugs from the perspectives of synchronization intentions," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 6, pp. 1060–1072, 2012.

[33] A. Tomb and C. Flanagan, "Detecting inconsistencies via universal reachability analysis," in *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, ser. ISSTA'12. New York, NY, USA: ACM, 2012, pp. 287–297.

[34] A. C. Nguyen and S.-C. Khoo, "Discovering complete api rules with mutation testing," in *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, 2012, pp. 151–160.

[35] J. Hoenicke, K. R. Leino, A. Podelski, M. Schäf, and T. Wies, "Doomed program points," *Form. Methods Syst. Des.*, vol. 37, no. 2-3, pp. 171–199, Dec. 2010.

[36] C. Csallner, Y. Smaragdakis, and T. Xie, "Dsd-crasher: A hybrid analysis tool for bug finding," *ACM Trans. Softw. Eng. Methodol.*, vol. 17, no. 2, pp. 8:1–8:37, May 2008.

[37] B. Chimdyalwar and S. Kumar, "Effective false positive filtering for evolving software," in *Proceedings of the 4th India Software Engineering Conference*, ser. ISEC'11. New York, NY, USA: ACM, 2011, pp. 103–106.

[38] H. Shen, J. Fang, and J. Zhao, "Efindbugs: Effective error ranking for findbugs," in *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*, 2011, pp. 299–308.

[39] A. Shi and G. Naumovich, “Field escape analysis for data confidentiality in java components,” in *Software Engineering Conference, 2007. APSEC 2007. 14th Asia-Pacific*, 2007, pp. 143–150.

[40] Y. Kim, J. Lee, H. Han, and K.-M. Choe, “Filtering false alarms of buffer overflow analysis using smt solvers,” *Inf. Softw. Technol.*, vol. 52, no. 2, pp. 210–219, Feb. 2010.

[41] E. Bodden, P. Lam, and L. Hendren, “Finding programming errors earlier by evaluating runtime monitors ahead-of-time,” in *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*, ser. SIGSOFT’08/FSE-16. New York, NY, USA: ACM, 2008, pp. 36–47.

[42] F. Otto and T. Moschny, “Finding synchronization defects in java programs: extended static analyses and code patterns,” in *Proceedings of the 1st international workshop on Multicore software engineering*, ser. IWMSE’08. New York, NY, USA: ACM, 2008, pp. 41–46.

[43] Q. Chen, L. Wang, and Z. Yang, “Heat: An integrated static and dynamic approach for thread escape analysis,” in *Computer Software and Applications Conference, 2009. COMPSAC’09. 33rd Annual IEEE International*, vol. 1, 2009, pp. 142–147.

[44] A. Shi and G. Naumovich, “Improving data integrity with a java mutability analysis.” in *APSEC*. IEEE Computer Society, 2007, pp. 135–142.

[45] A. Aggarwal and P. Jalote, “Integrating static and dynamic analysis for detecting vulnerabilities,” in *Computer Software and Applications Conference, 2006. COMPSAC’06. 30th Annual International*, vol. 1, 2006, pp. 343–350.

[46] D. Kong, Q. Zheng, C. Chen, J. Shuai, and M. Zhu, “Isa: a source code static vulnerability detection system based on data fusion,” in *Proceedings of the 2nd international conference on Scalable information systems*, ser. InfoScale ’07, 2007, pp. 55:1–55:7.

[47] M. G. Nanda, M. Gupta, S. Sinha, S. Chandra, D. Schmidt, and P. Balachandran, “Making defect-finding tools work for you,” in *Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 2*, ser. ICSE’10. New York, NY, USA: ACM, 2010, pp. 99–108.

[48] M. Al-Ameen, M. Hasan, and A. Hamid, “Making findbugs more powerful,” in *Software Engineering and Service Science (ICSESS), 2011 IEEE 2nd International Conference on*, 2011, pp. 705–708.

[49] C. Le Goues and W. Weimer, “Measuring code quality to improve specification mining,” *Software Engineering, IEEE Transactions on*, vol. 38, pp. 175–190, 2012.

[50] P. Anderson, “Measuring the value of static-analysis tool deployments,” *Security Privacy, IEEE*, vol. 10, pp. 40–47, 2012.

[51] S. Kim, K. Pan, and E. E. J. Whitehead, Jr., “Memories of bug fixes,” in *Proceedings of the 14th ACM SIGSOFT international symposium on Foundations of software engineering*, ser. SIGSOFT’06/FSE-14, 2006, pp. 35–45.

[52] C. Cifuentes and B. Scholz, “Parfait: designing a scalable bug checker,” in *Proceedings of the 2008 workshop on Static analysis*, ser. SAW’08, 2008, pp. 4–11.

[53] Z. Ding, H. Wang, and L. Ling, “Practical strategies to improve test efficiency,” *Tsinghua Science and Technology*, vol. 12, pp. 250–254, 2007.

[54] V. Pessanha, R. J. Dias, J. a. M. Lourenço, E. Farchi, and D. Sousa, “Practical verification of high-level dataraces in transactional memory programs,” in *Proceedings of the Workshop on Parallel and Distributed Systems: Testing, Analysis, and Debugging*, ser. PADTAD’11. New York, NY, USA: ACM, 2011, pp. 26–34.

[55] S. Heckman and L. Williams, “On establishing a benchmark for evaluating static analysis alert prioritization and classification techniques,” in *Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement*, ser. ESEM’08, 2008, pp. 41–50.

[56] S. Kim and M. D. Ernst, “Prioritizing warning categories by analyzing software history,” in *Proc. of Int’l Workshop on Mining Software Repositories (MSR’2007*, 2007, p. 27.

[57] K. Li, C. Reichenbach, C. Csallner, and Y. Smaragdakis, “Residual investigation: predictive and precise bug detection,” in *Proceedings of the 2012 International Symposium on Software Testing and Analysis*, ser. ISSTA’12. New York, NY, USA: ACM, 2012, pp. 298–308.

[58] D. Reimer, E. Schonberg, K. Srinivas, H. Srinivasan, B. Alpern, R. D. Johnson, A. Kershenbaum, and L. Koved, “Saber: smart analysis based error reduction,” *SIGSOFT Softw. Eng. Notes*, vol. 29, pp. 243–251, 2004.

[59] J. Wu, Y. Tang, G. Hu, H. Cui, and J. Yang, “Sound and precise analysis of parallel programs through schedule specialization,” in *Proceedings of the 33rd ACM SIGPLAN conference on Programming Language Design and Implementation*, ser. PLDI’12. New York, NY, USA: ACM, 2012, pp. 205–216.

[60] D. Babić, L. Martignoni, S. McCamant, and D. Song, “Statically-directed dynamic automated test generation,” in *Proceedings of the 2011 International Symposium on Software Testing and Analysis*, ser. ISSTA’11, New York, NY, USA, 2011, pp. 12–22.

[61] W. Han, M. Ren, S. Tian, L. Ding, and Y. He, “Static analysis of format string vulnerabilities,” in *Software and Network Engineering (SSNE), 2011 First ACIS International Symposium on*, 2011, pp. 122–127.

[62] W. H. K. Bester, C. P. Inggs, and W. C. Visser, “Test-case generation and bug-finding through symbolic execution,” in *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, ser. SAICSIT’12. New York, NY, USA: ACM, 2012, pp. 1–9.

[63] A. Avancini and M. Ceccato, “Towards security testing with taint analysis and genetic algorithms,” in *Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems*, ser. SESS’10. New York, NY, USA: ACM, 2010, pp. 65–71.

[64] S. Keul, “Tuning static data race analysis for automotive control software,” in *Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on*, 2011, pp. 45–54.

[65] N. Ayewah and W. Pugh, “Using checklists to review static analysis warnings,” in *Proceedings of the 2nd International Workshop on Defects in Large Software Systems: Held in conjunction with the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2009)*, ser. DEFECTS’09. New York, NY, USA: ACM, 2009, pp. 11–15.

[66] J. Lawall, J. Brunel, N. Palix, R. Hansen, H. Stuart, and G. Muller, “Wysiwib: A declarative approach to finding api protocols and bugs in linux code,” in *Dependable Systems Networks, 2009. DSN ’09. IEEE/IFIP International Conference on*, 2009, pp. 43–52.

[67] T. Kremenek and D. Engler, “Z-ranking: Using statistical analysis to counter the impact of static analysis approximations,” in *Static Analysis*, ser. Lecture Notes in Computer Science, R. Cousot, Ed. Springer Berlin Heidelberg, 2003, vol. 2694.

An Approach for Validation, Verification, and Model-based Testing of UML-based Real-time Systems

Mehdi Nobakht and Dragos Truscan
Department of Information Technologies, Åbo Akademi University, Turku, Finland
{mehdi.nobakht, dragos.truscan}@abo.fi

***Abstract*—UML is gaining popularity in designing real-time systems. However, UML tools often lack support for verification. This paper describes an approach and a tool in which UML models used for designing real-time systems are translated into UPPAAL timed automata in order to take advantage of validation and verification support in the UPPAAL tool. This allows one to increase the quality of the UML models by complementing static validation via OCL with behavioral validation and verification using the UPPAAL model-checker. Having an implementation of the system under consideration, the obtained UPPAAL timed automata serve as input of the UPPAAL-TRON tool to perform online model-based conformance testing. The proposed approach also generates a skeleton of the test adapter required to interface the testing tool and the implementation under test. The approach and the tool are exemplified with a telecommunication case study.**

***Keywords*—UML; UPPAAL; model verification; model-based conformance testing; real-time systems.**

I. Introduction

Unified Modeling Language (UML) [1] is a standardized general-purpose modeling language originally designed for the object-oriented paradigm. UML has also been suggested for designing embedded and real-time systems. It has been gaining popularity and is familiar to most designers and developers in this class of systems [2]. A key advantage of UML is the hierarchical mechanism giving a high degree of modularity and encapsulation to the model. It is particularly useful for modeling the behavior of complex systems. Moreover, an increasing number of UML tools provide code generation facilities which has increased its popularity further.

Once a real-time system is designed using UML, there is a need to ensure that the model conforms to the system specification. Model validation and verification methods aim at finding possible discrepancies between a system model and the corresponding specification at an early design stage. The Object Constraint Language (OCL) [3] is a formal language to supplement UML for detecting both syntactic inconsistencies and, to a limited extent, semantic ones in the models. While UML is particularly promising in designing embedded and real-time systems, it lacks support for verification of the timing and schedule related properties.

Testing is the pivotal part of real-time systems development process, being used to ensure that a product meets its requirements. This way, it helps to increase the quality of the product. *Model-Based Testing* (MBT) [4] is a testing technique which automatically generates tests from the behavioral specifications of the System Under Test (SUT). Depending on how tests are generated and executed, there are two flavors of MBT; in *offline testing*, the test cases are generated before the execution step, whereas through *online testing* both steps are integrated [5].

The work presented in this paper proposes an approach for validation, verification, and online model-based conformance testing of real-time systems which are designed using UML. In our approach, in order to compensate for the lack of formal and executable semantics of UML, the UML models including class and state machine diagrams are translated into the UPPAAL timed automata and later on validated and verified using the UPPAAL model-checker tool [6]. The translation is automated by a tool which beside creating the UPPAAL specifications, it propagates requirement information from the UML models to the UPPAAL timed automata and generates deadlock free and reachability queries for verification purposes. In addition, it generates a tester adapter stub required to interface UPPAAL-TRON – an online model-based testing tool [7] – and the Implementation Under Test (IUT).

Overview. The information presented in this paper will appear in the following order: Section II contains the works related to verification methods for the UML-based designs of real-time systems. Section III provides a background to the theory of timed automata, the semantics of timed automata as used by the UPPAAL toolbox, and the underlying principles of TRON. Section IV initially provides a UML solution for designing real-time systems explaining UML notation of static structures and timed state machines. Then, it describes the principles of our approach for translation of a UML model into UPPAAL timed automata and the tool support to automate the translation process. Section V describes a real telecommunication case study to demonstrate applicability of our approach. In addition, it describes the TRON test setup to perform model-based conformance testing. Section VI concludes the paper, while discussing future work.

II. Related Work

In the context of the UML model validation, Richters and Gogolla [8] propose the USE animation-based tool for validation of UML models and OCL constraints. We propose using the UPPAAL tool which integrates validation and verification processes. Later on, the obtained UPPAAL timed automata can also be used as input to the UPPAAL-TRON testing tool for test generation.

Work on verification of the UML-based design of real-time systems has been published by several authors. Similar to our approach, many of these authors base their approaches on the UPPAAL model checker and on the translation of UML to the input language of the UPPAAL, but in general they use different elements of UML.

A translation of the UML timed sequence diagrams into UPPAAL timed automata has been presented by Firley et al. [9]. Sequence diagrams specify required sequence of message between objects, but they are too weak to specify stronger properties like state invariants. In contrast, our approach uses class and state machine diagrams which are richer in expressing the system properties.

Similar to our work, Ober et al. [10] use class diagrams and state machine diagrams to capture the structure and the behavior of the system respectively. They utilize the IF toolset [11] to analyze the model and propose a translation from UML 1.4 to input language of IF, though no implementation of IF seems to be available. David et al. [12] suggest the time extension of the state diagram by adding clocks, timed guards, and invariants. However, their approach mainly focuses on flattening the hierarchical timed automata. Moreover, the event communication between processes has to be coded by hand.

A prototype tool called Hugo/RT has been presented by Knapp et al. [13]. It uses UML collaboration (sequence diagram) with time constraints and a set of timed UML state machines as input for the tool. However, their approach has several limitations. Most prominently, the input/output events between IUT and its environment model cannot have parameters. Muniz et al. [14] discussed an approach for verification of real-time systems represented for the CORBA component model. In their approach, UPPAAL is deployed for verification purposes and their tool called TANGRAM takes UML component and state machine diagrams to generate the equivalent UPPAAL timed automata. They extended the component diagram with a stereotype to model *event* passing between components. This mechanism does not allow to have parameterized events like [13]. Compared to these approaches, we use the UML interface element to model parameterized event passing.

III. Background To Timed Automata and UPPAAL

A. Timed Automata

According to theory of timed automata [15], a *timed automaton* is a non-deterministic finite state machine accompanied with *clock* to express timing properties. Clocks can be set to zero and their value increases linearly with time. At any instant, the value of a clock is equal to the time elapsed since the last time it was reset. The state of a system of timed automata includes the control state, variables and the clocks. Execution of timed automata are infinite sequences of system states that fulfil the invariants which may be either the passing of time or running of transitions. A transition is enabled either separately or synchronized with another automaton. The transition is taken when the associated time constraint is satisfied and its guard expression evaluates to true in the system state.

B. UPPAAL

UPPAAL is a tool-suite for modeling and model checking of real-time systems. It uses an extended version of timed automata, called UPPAAL Timed Automata (UPTA), to specify a system as a network of timed automata consisting *locations* and *transitions*. The behavior of the system is expressed by transitions (called edges in UPPAAL) between these locations. UPPAAL enriches the notion of *timed automata* by allowing to declare bounded integer variables in a automaton locally either or globally. Structured data types, user defined functions, binary channel synchronization, and broadcast channels are other UPTA extensions to timed automata. Moreover, it defines *urgent* and committed locations. In *urgent* location time is not allowed to pass as long as the location is active. Additionally, leaving the *committed* location has precedence over other possible transitions.

The channel synchronization between processes is denoted with $a?$ for the sending process, and with $a!$ for the receiving process. This way, several transitions are enabled simultaneously, but the assignment(s) in the *sending* automaton (with $a!$ label) is executed before the *receiving* automaton (with $a?$ label). This enables communication in a network of concurrent automata with the help of *global variables*. Value assignment and clock resetting can be two possible actions when a transition is enabled. It has to be noted that a transition is not taken when the resulting system state would not satisfy the associated invariant with the target location. The next system state is achieved by updating the control states of the timed automata involved in the transition by performing its defined actions. Furthermore, UPPAAL uses the idea of invariant which is a progress condition imposed on the location, that is, the system is not allowed to stay in the location more than the value mentioned by the invariant.

C. UPPAAL-TRON

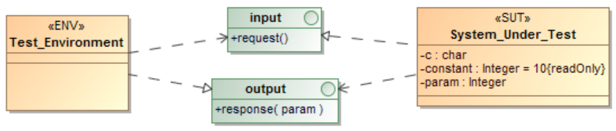
The UPPAAL-TRON tool – or simply TRON – is an extension to the UPPAAL tool for conformance testing of real-time systems, designed according to *relative timed input/output conformance relation (**rtioco**)* [5]. The test specification in TRON is partitioned into a model of the environment and a model of the SUT. These two models communicate using input/output channels. TRON attaches to the IUT via a *test adapter* which is a physical interface to enable communication between the testing tool and an implementation under test.

IV. Description of the Approach

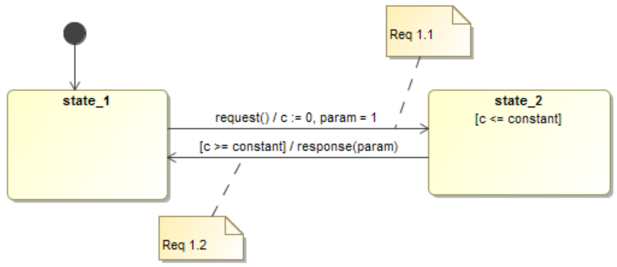
Throughout this section, we describe the main features of our approach to translate a UML model of a system, including *class diagram* and *state machines* into UPPAAL timed automata. More practical details and concrete examples can be found in [16].

A. UML Modeling

A real-time system interacts with its environment via input/output actions (from SUT's perspective). This work utilizes two types of UML diagrams to represent a real-time system and its environment: 1) a *class diagram*, describing SUT and its test system environment, and 2) the corresponding



(a) Class diagram specifying SUT and its environment


Req 1.1
state_1
request() / c := 0, param = 1
[c >= constant] / response(param)
state_2
[c <= constant]
Req 1.2

(b) State machine diagram describing behavior of SUT

Figure 1. Abstract UML model of a system.

state machine diagrams specifying the behavior of each class element.

The class diagram describes the entities involved in a test process: SUT and its environment testing system which are communicating using dedicated protocols. SUT is a real-time system taking input form the environment via communication networks and producing output to it. In our approach, class elements are used to represent all entities in a system which typically consists of SUT and its environment. We also deployed two model elements using *stereotype* extensibility mechanism in UML to distinguish SUT and its environment testing system in the system architecture model rendered as «SUT» and «ENV» respectively. The defined stereotypes are derived from the *base element* in UML. In our approach, all classes have to be stereotyped before proceeding to the translation into UPTA. We also allow for several classes to be stereotyped as SUT or as environment. At the testing time, the partitioning will be used for identifying the test interface.

We specify communication between entities via *interface* elements containing a set of operations. The interface specifies the operations which a given class (*supplier*) can provide to other classes (*clients*). The class can have attributes of type `integer` or `char`. The latter are used to define clocks in our UML model. Figure 1 shows an illustrative example of a system model including a SUT and its environment. The architecture of the system is depicted by the class diagram in Figure 1a, showing two class elements named *Test_Environment* and *System_Under_Test*. The *Test_Environment* sends a *request* message to *System_Under_Test* via *input* interface, and receives a *response* message accordingly.

Each class has associated state machine describing the behavior of the class element in terms of *states* and *transitions*. Figure 1b describes the dynamic behavior of SUT showing an *initial state* and two *simple states state_1* and *state_2*. A state can have *time invariant* specified as Boolean expression, (e.g., the SUT state machine is not allowed to stay in *state_2* more than *constant* time units after entering the state).

Events are triggers of transitions between states and response actions become *effects* on the transitions. In real-time systems, an event can be either *call event* or *time event* to trigger a transition. A fully defined transition includes a *trigger*, a *guard*, and an *action*. UML uses the following syntax for transitions:

```
event trigger(parameters)[guard]/action(s)
```

The guard condition is a Boolean expression which has to be met in order to fire the transition. The actions are executed only if the transition is taken. Transitions without any explicit trigger are triggered by an implicit *completion event* which occurs when all activities of the source state have been finished. In fact, it is handled like a time event with duration of 0 time units.

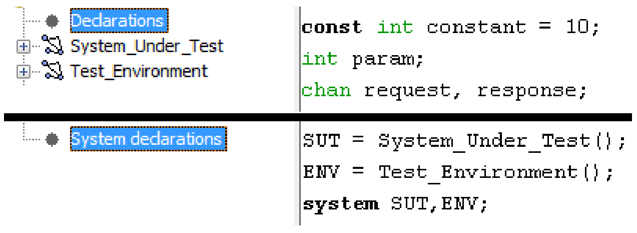
Specifying requirements. Requirements are modeled using SysML *requirements diagram* [17] and linked to different transitions in state machines, with the purpose of showing which requirements are fulfilled when a certain state is reached. An example of the approach can be found in [18]. For readability, in this paper, generic requirements are attached to transitions via a UML *comment* elements. For instance, in the state machine in Figure 1b, *Req 1.1* is achieved when the corresponding transition is taken and the state machine enters *state_2*.

B. *Translation from the UML model into UPTA*

A translation from UML models of real-time systems including class and state machine diagrams into UPTA consists of several steps as described below. Each step produces certain artifacts of UPTA.

1) Class element: class elements in class diagrams represent test entities in a test process. A class element in a UML class diagram whose behavior is defined by a state machine, is encoded by a timed automaton. Timed automata are represented by *templates* in UPPAAL. Templates are in turn instantiated to constitute the actual model.

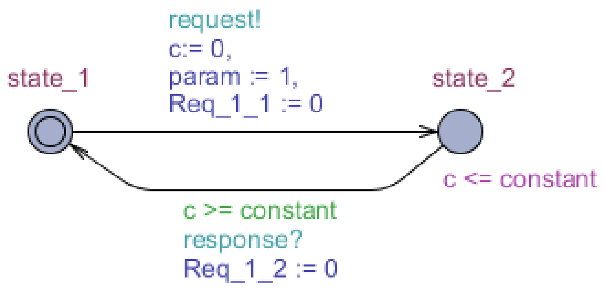
2) Interface and interface usage: A set of interface operations in the UML model is used as means of communication among test entities. The corresponding communication between templates in UPPAAL is represented by channel synchronizations. Each operation in an interface is translated into a binary synchronization channel in UPTA. The class element that realizes interfaces acts as the *receiving automaton*, whereas the class element that uses the interface acts as *sending automaton*. In addition, a list of interfaces in test adapter is created according to the interfaces between IUT and its environment. This list is used to generate Java source code including all input/output entries used by I/O handler as will be discussed later.



```
const int constant = 10;
int param;
chan request, response;
```

```
SUT = System_Under_Test();
ENV = Test_Environment();
system SUT,ENV;
```

(a) Declaration of SUT and its ENV



(b) SUT template

Figure 2. UPPAAL model of the example system.

3) Attributes: Class elements in UML class diagrams are inspected and for each attribute of integer type, a constant or an integer variable is declared in UPTA. For simplicity, all attributes with integer data type are declared globally in UPTA. UPPAAL only supports integer data types either as constants or variables. This approach takes advantage of this to represent `char` data type variables in class attributes as clocks. Consequently, these `char` variables are translated into the locally declared *clocks* of the corresponding timed automaton.

4) Superstates: In general, for each state in a UML state machine diagram, a single *location* is considered in a template. Initial and final pseudo-states in the UML state machine determine the initial and the final locations of the template, respectively.

5) Transitions: Each UML transition is represented by one or a sequence of *edges* in UPTA. Pertinent guards of a transition are copied appropriately to edge properties in UPTA. The trigger and effect actions of a transition are translated as receiving and sending binary synchronization channel respectively. In case a transition consists of a trigger and an effect action, it will be transformed by two *edges* and one *urgent location* in-between which the first edge is synchronized with the trigger and the second edge is synchronized with the effect action.

6) Requirements: Transitions in the UML model may have associated requirements. These requirements can be formulated as reachability properties and verified in UPPAAL. In addition, each requirement is translated into an auxiliary variable of type integer (initialized to 0) and attached on the corresponding edge in UPTA. These auxiliary variables are used during test generation for recognizing the coverage level or by formulating a property checking that an intended state can be reached or not.

7) Hierarchical state: UPPAAL does not support hierarchical locations. Thus, there is a need to flatten eventual hierarchical states in the UML state machines. This can be achieved by encoding hierarchical states as states of a flat timed automaton. Hierarchical states are replaced with several simple states so that the behavior of the system remains the same. Initial and final pseudo-states of sub-state machine are translated to *committed locations* in UPPAAL templates, and then, the transitions to and from sub-state machine are mapped to the corresponding committed locations.

C. *Tool support*

The transformation defined above is generic and can be used in conjunction with any UML based approach which follows the same modeling principles. To automate the transformation, a tool has been developed in Python as a MagicDraw [19] plug-in. As such, the transformation can be directly invoked from the GUI of MagicDraw and automatically produces equivalent UPTA and test adapter. An example of applying these transformations steps to the models in Figure 1 is shown in Figure 2.

V. The LTE Control-Plane Case Study

The applicability of our approach is demonstrated in a case study on the Long Term Evolution (LTE) [20] interface for cellular mobile telecommunication systems. The LTE network consists of the access network and the core network. Evolved Universal Terrestrial Radio Access Network (E-UTRAN) is the radio access network technology and Evolved Packet Core (EPC) is the core part. Together, they form the Evolved Packet System (EPS). The EPC consists of Packet Data Network Gateway (PDN-GW) router and Mobility Management Entity Serving Gateway (MME/S-GW) router. The latter is split into two parts: Mobility Management Entity (MME) – managing the control plane and tracking user equipment; and Serving Gateway (S-GW) – dealing with user plane IP packets. The E-UTRAN NodeB (eNodeB) network element is a central

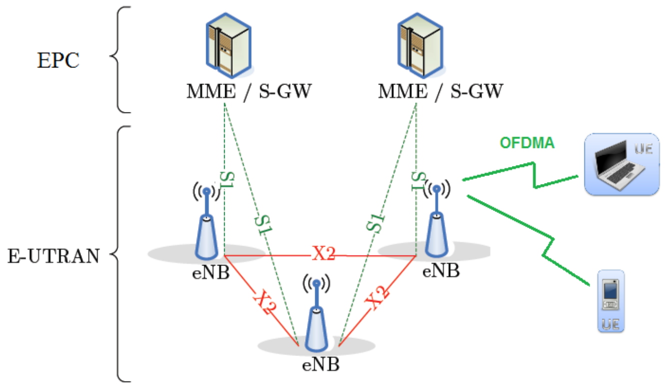
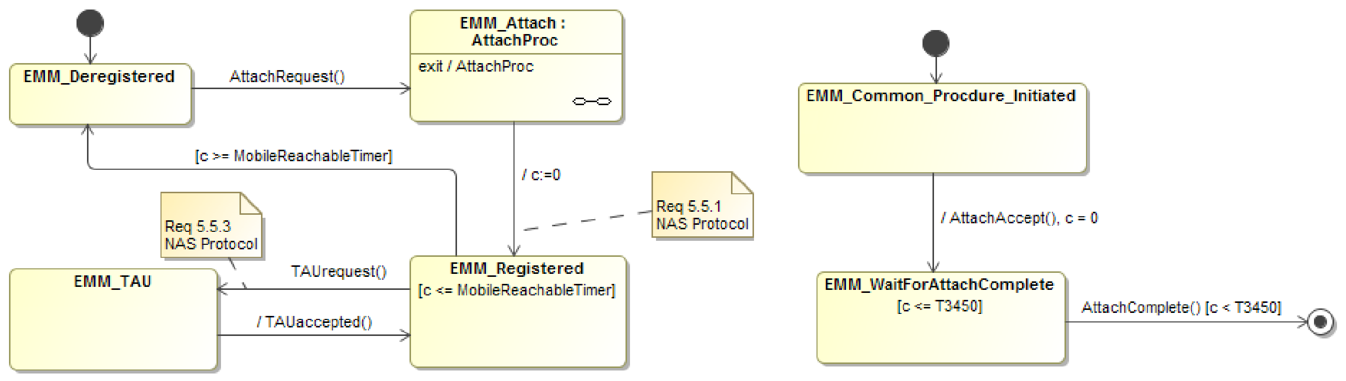


Figure 3. LTE Overall Architecture [20].



(a) The MME state machine

(b) The MME attach sub-state machine

Figure 4. Dynamic Behavior of MME.

network element in the LTE infrastructure whose main functionality is to connect a User Equipment (UE) (e.g., a mobile phone) to the MME. The interface between the UE and the eNodeB is a radio interface, while the interface between eNodeB and MME, called S1AP, often is a fiber optic; refer to Figure 3.

Here, the main focus is on specific parts of the LTE control plane focusing on *Initial Attach* and *Tracking Area Updating* procedures from the EPS Mobility Management (EMM) layer from the None Access Stratum (NAS) protocols [21]. NAS is the highest stratum of the control-plane between the UE and the MME accounting for *mobility management* and *session management*. We designed a UML model to reflect the structure of UE and MME, and to express the behavior of aforementioned procedures, which are represented by class and state machine diagrams respectively. Implementations of the UE and of the MME were developed according to the UML model; however, only the MME will be used as SUT in this paper.

The main goals of the case study are: 1) to validate and verify the UML models of these two procedures regarding the NAS protocol requirements using the UPPAAL tool, and 2) to perform timed model-based conformance testing against the implementation of MME using TRON in order to determine whether the implementation conforms to the models.

A. EMM specific procedures

The *Initial Attach procedure* creates UE context when a UE is turned on and attaches to the network. According to Section 5.5.1 of the NAS Protocols, the UE sends a NAS *Attach Request* message to the MME via the eNodeB, starts timer T3410. The *Attach Request* reception in the MME is acknowledged with *Attach Accept* message and followed by starting timer T3450. Reception of the *Attach Accept* message by the UE causes to stop timer T3410. If timer T3410 expires prior to receiving an *Attach Accept*, the attach procedure is restarted. The MME also triggers the *update location* procedure, as well as the *route establishment* procedure. It communicates with Home Subscriber Server (HSS) and Home Location Register (HLR) in the update location procedure. S-GW is another entity that the MME communicates with it for route establishment procedure. After the bearers in the core network have been established, The MME tries to establish user-plane transport functions on interface between the UE and the eNodeB, as well as interface between the eNodeB and the MME. After establishment of user-plane, the UE sends *Attach Complete* message to the MME in order to confirm the assignment of user-plane tunnel. The MME supervises the reception of the *Attach Complete* by T3450 timer. However, in this case study, our main focus is on NAS protocols between MME and UE, making eNodeB, HSS, HLR, and S-GW irrelevant.

Based on Section 5.5.3 of the NAS Protocols, the UE must periodically perform tracking area updates procedure in order to update the registration of its actual tracking area in the network. This procedure is controlled in the UE by means of timer T3412. When timer T3412 expires, the tracking area update is started by sending *Tracking Area Update Request* to the MME. If this request has been accepted by the network, the MME shall send a *Tracking Area Update Accept* to the UE. The MME supervises the periodic tracking area updating procedure of the UE by mobile reachable timer which according to the protocol is 4 minutes greater than timer T3412. Upon expiry of the mobile reachable timer, the MME considers the UE to be inactive and performs *Detach procedure* to cancel the registration of this particular UE.

B. UML models for SUT and the environment

Here, we assume that the MME acts as SUT and the UE as its environment. However, having the implementation of both entities allows changing their role. Figure 4 displays the UML model for behavior of MME to support *initial attach* and *tracking area updating* procedures. The MME model is designed according to the procedures defined in the NAS protocols specification as explained earlier. The state machine of the MME in Figure 4a shows a hierarchical state named *EMM_Attach*. This gives modularity to the model and makes it easier to follow. The sub-state machine itself consists of one initial state, one final state, and two simple states, as presented in Figure 4b. The comment elements on the MME state machine named `Req 5.5.1` and `Req 5.5.3` express clearly the satisfying condition for the *initial attach* and *tracking area updating* procedures respectively.

C. Generating the UPPAAL model

Once the UML model of MME and UE includes all the necessary elements, it serves as input of the transformation tool to generate an equivalent UPTA using our tool. The resulting MME automaton in Figure 5 corresponds to the MME state machine in Figure 4 and exhibits the same behavior. It is worth mentioning that the hierarchical state machine in Figure 4 has been flattened automatically by the tool and included in the automaton of its parent.

1) Validation: The simulator tab of UPPAAL allows exploring the UPTA in a guided or random fashion without being exhaustive. When the simulation tab is selected, prior to the simulation phase, UPPAAL performs *syntax checking* which validates the UPTA with regard to *consistency*, *correctness*, and *completeness*. Once the syntax checking has succeeded, the UPPAAL simulator allows following the execution of the models visually, checking the instantaneous states and variables, and inspecting the communication trace between the UE and the MME parallel processes.

2) Verification: Different properties of the resulting model can be verified in UPPAAL. These properties are specified as queries written using a simplified version of Timed Computation Tree Logic (TCTL) [22]. The UPPAAL query language consists of the path and the state formulas. The path formulas quantify the paths or the traces of a UPTA with temporal logic, while state formulas describe individual states with regular logical operators.

As mentioned in the previous sections, our UML to UPTA translation automatically creates two types of queries. Firstly, we generate 'no deadlock' query to facilitate checking of this property in the system model.

```
A[] no deadlock
```

Secondly, we generate queries for checking the *reachability* property for the states whose incoming transition are tagged with the *comment element*. In our case study, the following query was produced by our tool, according to *Req 5.5.1* in Figure 4, and used by UPPAAL verifier to check whether the MME automaton eventually reaches the location *EMM_Registered*.

```
E<> MME.EMM_Registered
```

However, the *reachability* property does not guarantee the correctness of a system model, i.e., it just checks the basic behavior of the system model by performing such *sanity* checks. For instance, when the MME automaton enters the location *EMM_Deregistered* after the registration of a UE, the *mobile reachable* timer must have been expired. This requirement can be expressed with the following *safety* property:

```
A[] MME.EMM_Deregistered imply
                MME.c >= MobileReachableTimer
```

D. TRON Test setup

The test setup for the MME entity of LTE includes TRON engine and its internal Socket Adapter, the TCP/IP Socket with input/output handler, and an implementation of MME as shown in Figure 6. The I/O Handler translates abstract inputs from TRON into concrete physical actions for the IUT. On the other hand, it recognizes physical output of the IUT and then encodes it into proper abstract message readable by TRON. The I/O Handler communicates with the TCP/IP Socket and the IUT via function calls. Communication between TRON built-in adapter and MME is done via TCP/IP.

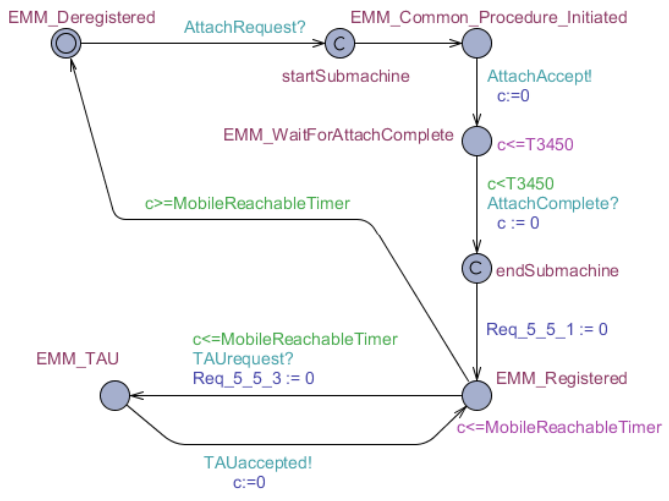


Figure 5. The MME UPPAAL Template.

Inputs in the implementation model are *AttachRequest*, *AttachComplete*, and *TAUrequest* and outputs correspond to *AttachAccept* and *TAUaccepted*, refer to Figure 5. TRON derives test cases directly from the environment model by choosing one of the possible inputs within allowed time delay at each state using the UPPAAL engine. It then executes them against an IUT and observes the output. Finally, it evaluates the correctness of a test experiment based on the model of IUT and determines the test verdict. Since TRON is an online testing tool, it keeps the connection to the IUT in real-time when performing all of the test procedure steps.

VI. Conclusions and Future Work

The proposed approach is aimed at increasing the quality of UML-based models of real-time systems via validation and verification using UPPAAL. For this purpose, we suggested an approach in which UML specifications are created and subsequently transformed into UPPAAL timed automata. Whenever a problem is discovered in the UPTA specifications, the UML model is updated and then re-transformed. Using this approach allows using UML and UPTA in a complementary fashion.

At UML level, our approach allows one to clearly identify the SUT and the test environment and to model their behavior and the communication interfaces. Via a set of mappings, we translate these models into UPTA. The translation also propagates requirement-related information which is then used to generate reachability properties.

The resulting UPTA specifications are also used for test generation using the TRON tool, which allows for generating and executing tests in timely fashion. One overhead of setting up the online MBT toolchain is the creation of the test adapter, which requires an initial investment followed by relatively

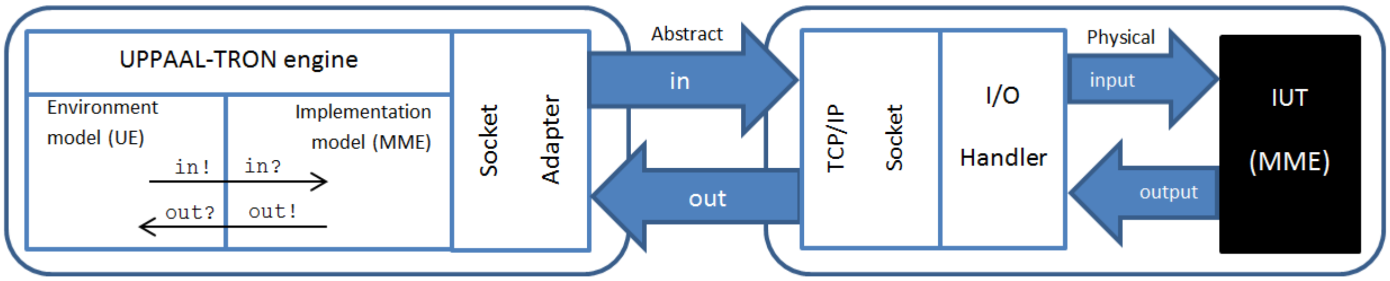


Figure 6. Specific TRON setup.

small updates each time the interfaces of the SUT is updated. In order to cut down on this initial investment, we generate a skeleton of the adapter during the transformation as described in [16]. Using TRON for model-based conformance testing, we managed to uncover a number of bugs in the implementation of MME which were addressed accordingly.

One current limitation of our approach is its scalability. Increasing the complexity of the specifications may result in a state space explosion in UPPAAL during verification and test generation. Although some ad-hoc optimizations can be considered to avoid this problem, we plan to search for a more systematic approach in future work.

In this study, we restricted ourselves to a limited set of UML model and extend this with real-time elements such as clock and state invariant. The clock expression in the UML state machine using the `char` data type is rather limited. Further research in this context will look into a more elaborated modeling of time and clock in UML. In addition, we will investigate how more UML diagram types can be included in our approach.

References

[1] (2013, August) Documents associated with unified modeling language (UML), version 2.4.1. [Online]. Available: http://www.omg.org/spec/UML/2.4.1/

[2] L. Lavagno, G. Martin, and B. V. Selic, *UML for Real: Design of Embedded Real-Time Systems*. Secaucus, NJ, USA: Springer, 2003.

[3] (2013, August) Documents associated with object constraint language (OCL), version 2.3.1. [Online]. Available: http://www.omg.org/spec/OCL/2.3.1/

[4] M. Utting, "The role of model-based testing," in *Verified Software: Theories, Tools, Experiments*, ser. LNCS. Springer, 2008, vol. 4171, pp. 510 – 517.

[5] A. Hessel, K. G. Larsen, M. Mikucionis, B. Nielsen, P. Pettersson, and A. Skou, "Testing real-time systems using UPPAAL," in *Formal Methods and Testing*, ser. LNCS. Springer, 2008, vol. 4949, pp. 77–117.

[6] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *Formal Methods for the Design of Real-Time Systems*, ser. LNCS. Springer, 2004, vol. 3185, pp. 200–236.

[7] K. G. Larsen, M. Mikucionis, B. Nielsen, and A. Skou, "Testing real-time embedded software using UPPAAL-TRON: An industrial case study," in *Proc. 5th ACM international conference on Embedded software*, Jeresy, NJ, USA, September 2005, pp. 299–306.

[8] M. Richters and M. Gogolla, "Validating UML models and OCL constraints," in *«UML» 2000 - The Unified Modeling Language*, ser. LNCS. Springer, 2000, vol. 1939, pp. 265–277.

[9] T. Firley, M. Huhn, K. Diethers, T. Gehrke, and U. Goltz, "Timed sequence diagrams and tool-based analysis - a case study," in *«UML» '99 - The Unified Modeling Language*, ser. LNCS. Springer, 1999, vol. 1723, pp. 645–660.

[10] I. Ober, S. Graf, and I. Ober, "Validating timed UML models by simulation and verification," *International Journal on Software Tools Technology Transfer*, vol. 8, no. 2, pp. 128–145, 2006.

[11] M. Bozga, J.-C. Fernandez, L. Ghirvu, S. Graf, J.-P. Krimm, and L. Mounier, "IF: An intermediate representation and validation environment for timed asynchronous systems," in *FM '99 - Formal Methods*, ser. LNCS. Springer, 1999, vol. 1708, pp. 307–327.

[12] A. David, M. O. Möller, and W. Yi, "Formal verification of UML statecharts with real-time extensions," in *Fundamental Approaches to Software Engineering*, ser. LNCS. Springer, 2002, vol. 2306, pp. 218–232.

[13] A. Knapp, S. Merz, and R. Christopher, "Model checking - timed UML state machines and collaborations," in *Proc. 7th International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*, Oldenburg, Germany, September 2002, pp. 395–416.

[14] A. L. N. Muniz, A. M. S. Andrade, and G. Lima, "Integrating UML and UPPAAL for designing, specifying and verifying component-based real-time systems," *Innovatioin in Systems and Software Engineering*, vol. 6, no. 1-2, pp. 29–37, 2010.

[15] R. Alur and L. D. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, no. 2, pp. 183–235, 1994.

[16] M. Nobakht and D. Truscan, "Tool support for transforming UML-based specifications to UPPAAL timed automata," Turku Centre for Computer Science (TUCS), Tech. Rep. 1087, June 2013. [Online]. Available: http://tucs.fi/publications/view/?pub_id=tNoTr13a

[17] (2013, August) Documents associated with systems modeling language (SysML), version 1.3. [Online]. Available: http://www.omg.org/spec/SysML/1.3/

[18] F. Abbors, D. Truscan, and J. Lilius, "Tracing requirements in a model-based testing approach," in *Proc. First International Conference on Advances in System Testing and Validation Lifecycle*. Porto, Portugal: IEEE Computer Society, September 2009, pp. 123–128.

[19] (2013, August) MagicDraw webpage on NoMagic. [Online]. Available: http://www.nomagic.com/products/magicdraw/

[20] *ETSI TS 136 300 Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access (E-UTRAN); Overall description; Stage 2*, ETSI Std., Rev. V8.4.0, 04 2008.

[21] *ETSI TS 124 301 Universal Mobile Telecommunications System (UMTS); LTE; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3*, ETSI Std., Rev. V8.10.0, 06 2011.

[22] R. Alur, C. Courcoubetis, and D. Dill, "Model-checking for real-time systems," in *Proc. Fifth Annual IEEE Symposium on Logic in Computer Science*, ser. LICS '90, 1990, pp. 414–425.

Toward a Definition of π-DSL for Modelling Business Agents

MDA based π–calculus extension

Charif Mahmoudi and Fabrice Mourlin
Laboratory of Algorithms, Complexity and Logics,
Paris 12th University
Créteil, France
{charif.mahmoudi, fabrice.mourlin}@u-pec.fr

***Abstract*—In this paper, we will address the issue of modeling the integration of agents with various resources and services, as found in an Service-Oriented Architecture (SOA) platform. We are proposing an approach for modeling agents and integrating these agents in existing pipes and filters based message routing and mediation engines. Using Model-driven development (MDA) as a base for our modeling strategy, our agent model generates source code based on Enterprise integration patterns (EIP) by Hohpe and Woolf. We are presenting a new agent design that uses the Open Gateway Services Interfaces (OSGi) architecture as an agent platform and the Apache Camel enterprise integration framework as the EIP based engine. The approach is illustrated by a business process use case, and a complete example including process specification and code generation. The main objective of the example is to demonstrate the benefits of using agents as orchestration of external services via a specialized message routing engine that supports EIPs.**

Keywords- Process algebra; Orchestration languages; Software agents; Web services; EIP; π–DSL; MDA; SOA; OSGi

I. INTRODUCTION

In the business world, the orchestration of Web Services is becoming increasingly widespread [1] This technology allows, via tools, a simple way to handle graphically different business needs. We give as an example BPMN [2]. Other specifications can be described as the specification for the construction of orchestrations as Apache CAMEL [3] and Spring Integration [4]. For some researchers [5], the specifications based on based on Enterprise Integration Pattern (EIP) [6] are dedicated routing within ESB [7]. But most of them [8] agree that specifications based on EIP are ideal for building orchestrations. In addition, it should be noted that most of the specifications based on EIP do not offer graphical tools to develop visually unlike BPMN specification.

In this paper, we will present an approach allowing orchestrations in a mobile agent [9] form based on the EIP specifications. This approach is based on the work [10] that we previously published and which we consider as the foundation of an OSGi [11] based ecosystem able to run mobile agents.

The paper is organized as following. We review a number of related works in Section 2, and describe the standards we have set as a framework of our work in Section 3. Section 4 provides the detail of the MDA approach that we used to define our system. Section 5 presents the formal specifications of our EIP based target system. It uses EIP specifications as a mean to declare a mobile orchestration carrying agent [12]. We conclude our work, and describe the future work in Section 6.

II. WORK CONTEXT

In the context of SOA [13], the orchestration has a central role since it defines the steps to be performed to provide a result. The steps are Web services calls, the results of the various services are handled by the orchestrator. The final result of the orchestration is based on the results of each step.

The orchestrations are defined by the W3C (glossary) as "the pattern of interactions that must respect a Web service agent to achieve its purpose." Based on this definition, we can consider an orchestration as a director of a software agent (program) behavior [14]. The agent exposes a Web service that is available to other agents, the result returned by the agent consists of a series of calls to basic services and transformations on the data retrieved from the basic services used. Figure 1 illustrates a simple agent based orchestrations [15].

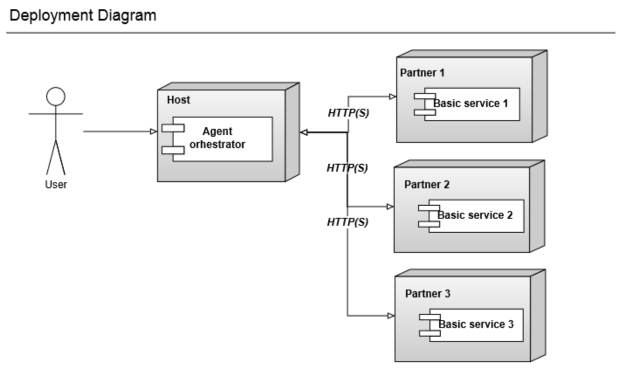


Figure 1. Connections of orchstration.

An orchestration gives rise to a semantic once interpreted. The benefit of orchestration is noticed during interpretation. The same semantics can come from many styles of definitions. The model depends on the language used to implement the definition of an orchestration.

The approach that we present is an EIP based model of orchestration definition. The proposed model allows the building of orchestrations with semantics quite similar to those built by other models in the domain [16] [17] [18].

A. *Business Agent*

Our approach allows managing orchestrations composed of EIP's. In this section, we will see what a software agent; we will also see how to use an orchestration within an agent.

A business agent is an agent first. In addition, this agent assures the autonomy property. An agent is a program that is autonomous [19]. It has the ability to communicate with its environment and to perform the task for which it was made.

An agent is characterized by four main features:

• Autonomy: an agent is master of its decisions. Its behavior is not directed from the outside but it is self-managing agent. We can see the property of autonomy in two aspects: autonomy of the internal state of the agent and the autonomy of the agent's actions. Internal autonomy means that the agent is able to change its state by objective. The autonomy of action means that the agent is able to make a decision based on the information from its environment. Both aspects of the autonomy of the agent are provided by the π-DSL language. The ultimate goal of the agent is to compose a response to an invocation. This composition is based on communications with business and monitoring components.

• Reactivity: the agent is able to perceive the changes in its environment using the components of monitoring and possibly take action in response to changes in this environment.

• Proactivity: an agent is able to determine the actions to achieve its objective, it is based on its internal state and the information received from its runtime environment.

• Social: an agent is able to communicate with other agents, to carry out its mission and achieve its objective. Given that agents expose their services using the same interface type as the components business. Calls to agents and business services base happens in a transparent manner.

A business agent is a composition of business services characterized by four properties of the agent. These four properties are provided by our approach to defining business agent A. π-DSL.

B. *EIP orchestration*

Several EIP based specifications exist, which were not initially dedicated to Web services orchestration, but could be used as tools allowing orchestration, like Translator or Aggregator. We have decided to base our approach on these specifications. These EIP specifications are the base of the different interactions with basic services as well as the transformations necessary to build an orchestration. Thereby, orchestrating inherits the properties of the EIP that compose it. Note that the order of definition is important and must be preserved during execution.

EIPs provide a framework for interacting with partners to transform the data flow and be invoked by other partners. Each EIP provides a work step, i.e., interaction in the orchestration; it is possible to have a work step composed of several EIPs.

Given that the EIPs are based on the "pipe and filter" architecture, they automatically provide the concepts of channel messages, routing, transformation and endpoint. Messages are what travel between a pipe and a filter. The structure of a message is as specified in the JMS [20]. In this paper, a channel allows a message to transit and an endpoint is a destination of the message. In addition, EIPs introduce the concepts of routing and transformations between channels and endpoints.

Our orchestration will be a composition in which each step is based on one or more of EIP concepts.

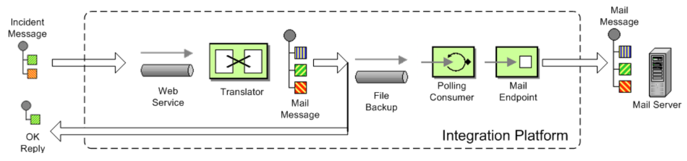


Figure 2. An EIP based system

Figure 2 shows some EIPs, and how it is possible to build an EIP from basic treatments. These treatments are basic bricks we use to define our orchestrations.

Our orchestrations are exposed as Web Services endpoints. When an exposed endpoint is invoked, the orchestration activates the different EIP component of the requested orchestration. Activation of an orchestration can allow data transformation, invoking the participants in this orchestration and returning a result to the client on the initiative of the invocation on the exposed endpoint.

Our system supports various treatments and activities offered by other systems, such as BPM orchestration. The difference lies in the fact that the treatments and activities are implemented within well-defined patterns.

III. FORMAL SPECIFICATIONS

In this section, we will present the formal specifications of our system. We will start by a reminder of the π-calculus language, then we will present and comment on some parts of the specifications of our system and finally, we will present an example of agent-based orchestration definition as a foundation of our case study.

A. *π-calculus*

The π-calculus is a formal language designed to define concurrent systems. The language basically focuses on the communication between parallel systems. The language was developed by R. Milner [21] and was published for the first time in [22]. The π-calculus is based on the concept of terms and names. Term represents a process or sub-process. Also, a term consists of a sequence of emissions and receptions via communication channels. It also consists of calls to other terms. However, a name can be either a communication channel or a variable that will be calculated by the values received via a channel.

$$\begin{aligned} S &\overset{\text{def}}{=} (\nu\, c\, d)\, (P(c,d) | Q(c,d)) \\ P(c,d) &\overset{\text{def}}{=} (\nu\, b)\, \big(c(a).\tau.\bar{d}\langle b\rangle \big| \emptyset\big) \\ Q(c,d) &\overset{\text{def}}{=} (\nu\, a)\, (\bar{c}\langle a\rangle.\tau.d(b) | \emptyset) \end{aligned} \quad (1)$$

The equation (1) is a definition of *S*, a term that execute in parallel the term *P* and *Q* that use the canals *c* and *d* to communicate with each other. This definition is expressed using one of the three variations of the π-calculus, which is the monadic π-calculus. This variation characteristic is that a communication channel can transfer only a single value.

The second variation of the π-calculus is polyadic π-calculus. The main difference between the monadic and polyadic is that the latter can transmit and receive multiple names on the same channel as demonstrated in the (2) using the same example from term "S".

$$\begin{aligned} S &\overset{\text{def}}{=} (\nu\, c)\, \big(P(c) \big| Q(c)\big) \\ P(c) &\overset{\text{def}}{=} (\nu\, b)\, \big(c(a, callback).\tau.\overline{callback}\langle b\rangle \big| \emptyset\big) \\ Q(c) &\overset{\text{def}}{=} (\nu\, a\, callback) \\ &\quad (\bar{c}\langle a, callback\rangle.\tau.callback(b) | \emptyset) \end{aligned} \quad (2)$$

The third variation is the π-calculus of higher order. This variation contains all the characteristics of the polyadic π-calculus. In addition, it allows to send and receive terms and names via a channel in the same way. The equation (3) shows the transfer of a term 'R' between terms 'P' and 'Q'. Therefore, showing that the execution of the term 'R' is on the target process.

$$\begin{aligned} S &\overset{\text{def}}{=} (\nu\, c)\, (P(c) | Q(c)) \\ R(callback, param) &\overset{\text{def}}{=} \\ &\quad (\nu\, b\,)\, \big(param(v).\tau.\overline{callback}\langle v\rangle \big| \emptyset\big) \\ P(c) &\overset{\text{def}}{=} (\nu\, b) \\ &\quad (c(a, param, Process).\tau.\overline{param}\langle b\rangle | Process) \\ Q(c) &\overset{\text{def}}{=} (\nu\, a\, callback\, param) \\ &\quad (\bar{c}\langle a, param, R(\, callback, param)\rangle.\tau.callback(b) | \emptyset) \end{aligned} \quad (3)$$

We will use the extension communication operator [23] in a polyadic context as shown below:

$$x.F \mid \bar{x}.C \;\; \rightarrow F \bullet C \quad (4)$$

Let us define the following:

$$\begin{aligned} F &\overset{\text{def}}{=} (\lambda\, \vec{y})P \\ C &\overset{\text{def}}{=} [\vec{y}]Q \end{aligned} \quad (5)$$

The operator $-\bullet-$ allows us to define an interface between the two terms in which it operates. This will make possible to dynamically integrate terms with the entire orchestration steps. This operator can be assimilated to a communication interface in UML as shown in Figure 3.

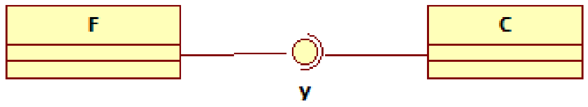


Figure 3. π-calculus interface

B. *Construction of a definition of orchestration*

We consider 'Orch' an orchestration with a single participant. The variable IN from (6) represents an input of the orchestration:

$$IN \overset{\text{def}}{=} (\nu\, y)[\vec{y}] \mid (\lambda\, \vec{z}).\tau \quad (6)$$

And the term OUT in (7) is the sole participant in the orchestration:

$$OUT \overset{\text{def}}{=} (\lambda\, \vec{z}).\tau \mid \; [\vec{y}] \quad (7)$$

The vector $\overrightarrow{Pr}$ represents all the terms corresponding to processing steps and transformations performed between receiving a request and returning the result. We can then define the term 'Orch' as follows:

$$\text{Orch} \overset{\text{def}}{=} IN \bullet \Big(\big((\lambda\, \vec{y})\; Pr_i\; [\vec{z}]\,\big) \bullet\Big)^{\|Pr\|} OUT \quad (8)$$

The term „Orch" given in (8) creates a flow through all terms Pr_i between the input 'IN' and the output 'OUT'. Each term Pr_i representing a step in the orchestration will have a vector of names as input. Each term will have a second vector Pr_i as output. These vectors will be transported between the different steps following the same order defined within the vector $\overrightarrow{Pr}$. The input $\vec{y}$ to the Term Pr_i is connected to the output $\vec{z}$ of the term Pr_{i-1} while its output is connected to $\vec{z}$ the input $\lambda\, \vec{y}$ of the term $Pr_{i\pm 1}$.

The operator "•" is an ideal way to represent an exchange that carries the communication streams between two steps of an orchestration. This operator will help us to connect the various processes that define an orchestration.

As we have seen, our orchestrations are in the form of a set of steps (transformations) between an endpoint and the participants of the orchestration. The list of steps has not been known by the engine before loading the definition of orchestration. We will use a data structure in order to persist the definition of orchestration. The instance of this structure will be loaded by the engine via an activator that is a particular endpoint type for connecting managed services to an input channel. The engine will be based on this definition that it receives in the form of a linked structure to activate the orchestration.

Activation of the orchestration can link the different steps. As illustrated in Figure 4, the link between these steps is the connection of inlet flow of step 'n' with the exit of 'n-1' using the concept of exchange, which carries a two-way flow.

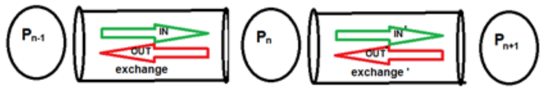


Figure 4. An EIP based system

We will use both π-calculus concepts of abstraction and concretion in order to implement dynamic linking on chained lists. These lists will be used to contain the different steps of our orchestration.

IV. AN APPROACH BASED ON MDA

We defined the π-calculus language as meta-meta-model. In Section 5, we will present the definition of a meta-model in π-calculus. Meta-model consists of an extension of π-calculus as dedicated to DSL service orchestration based routes. Routes are an implementation of pipe and filter architecture using routing rules. The proposed DSL takes a form of a composition of EIP. Meta-model also describes the tools needed to run a model once created. These tools are in the form of a set of components. The models are created using the π-calculus based DSL. Figure 5 illustrates the four levels of our approach.

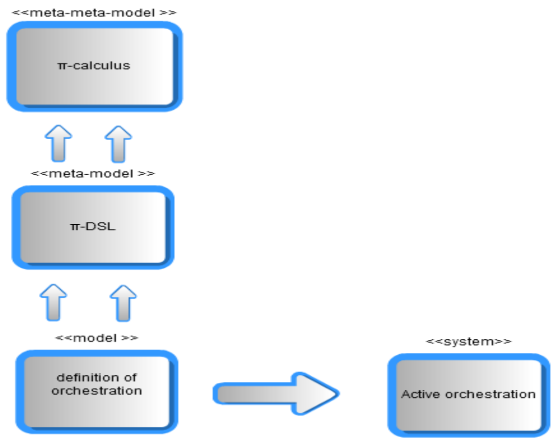


Figure 5. MDA Model

In the next section, we will detail the transformations made between the different models.

A. Model-driven orchestrations definition

Our approach in defining orchestrations is a MDA based approach [24]. The business area of our system is the definition of orchestrations; these orchestrations are components of the fundamental services. We have extracted domain-specific vocabulary as a π-DSL language. We can represent the π-DSL as a set of terms called EIP when EIP = {from, process, to ...}

Each orchestration will be defined using a language described in π-calculus. This language allows the interaction between various tools made available to the orchestrations.

Our meta-meta-model describes a language of orchestration in addition to the tool permitting the interpretation of this DSL orchestration language. The interpretation tools using π-DSL will be subject to a manual transformation [25] to object-oriented programming language [26]. The execution of the system supports different terms materialized from meta-meta-model in order to connect via the EIP channels. These channels are essential to the π-DSL.

Each orchestration is defined as a set of "emissions" on the EIP channels. Emissions existing on the EIP channels are received by one of the tools, which are the same as the term Routes that will be described in detail in subsequent section. We will also specify the term Route that allows transforming the definition of a π-DSL orchestration into a definition taking the form of data structure. This data structure represents the Platform independent model (PIM) [27] orchestration.

The structure representing the PIM is transformed in order to activate the orchestration. The step involving the activation transforms the structure representing the PIM in an executable code representing an orchestration language. The code will be generated automatically as Camel java-DSL [28]. The Camel DSL code communicates on the same channels as the EIP tools defined in the meta-meta-model.

Figure 6 illustrates an example of an orchestration that uses a service that transforms the Route of this service before returning it to the customer at the initiative of the invocation. Consumer and Provider are specific process wrappers for external endpoints interaction.

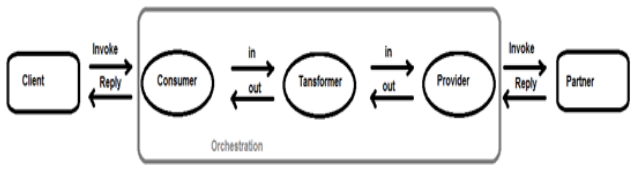


Figure 6. Exchages in orchestration

Our goal is to reach an executable system from the definition in the form of π-DSL. To do this, we perform a set of transformations whose outlines are highlighted in the Figure 7.

In the next section, we detail the structure of meta-meta-model orchestrations then in the next section, we will talk about the definition of the various EIP, which constitute the π-DSL routing and orchestration oriented language. Then, in the section dedicated to message route, we will detail the activation principle such as we designing our approach.

B. Model-driven orchestrations transformations

In our approach, the definition of orchestrations is the body of the wrapper agent of these orchestrations. Each agent has a definition, which characterizes it by an orchestration that is unique for the agent itself. Applying the definition of the agent in our system triggers a change in the system state. This new state is reached after the activation of the orchestration definition. The activation implements the semantics described by the definition of orchestration.

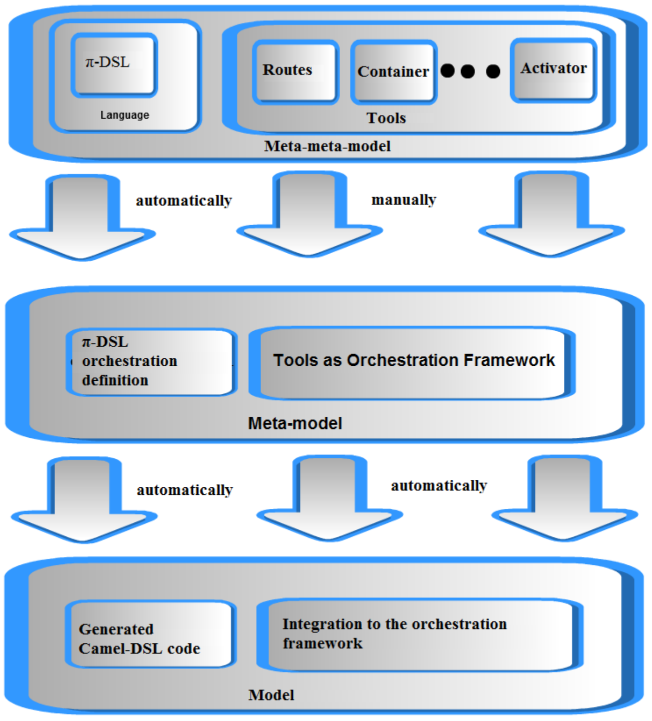


Figure 7. MDA transformations levels

Orchestrations will use the concept of route introduced by EIP. The Route is the building blocks of an orchestration. The Route is used to associate an input to transformations and outputs. Inputs are endpoints exposed by the agent while the outputs are endpoints consumed by the agent. Transformations can be applied to both input and output stream flows.

The Figure 8 shows an orchestration using the content based router EIP and message translator EIP to route the input message to the adequate translator

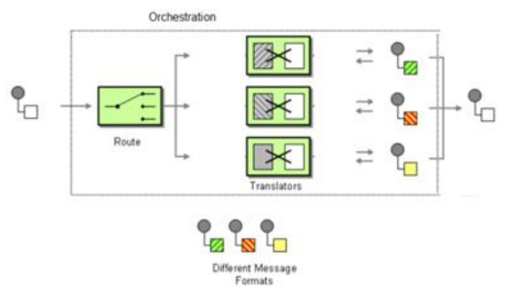


Figure 8. An EIP orchestration

The definition of an orchestration and the semantics of an orchestration are separate concepts. So far we have only discussed the definition of orchestration, which is composed of the series of actions to take in response to an external invocation. Each orchestration is a model. It is described using the π-DSL, which is the extension of π-calculus offered by the meta-meta-model (see Section 5).

The π-DSL consists of all the EIP channel names. It defines an orchestration through signals on EIP names. Since π-DSL is an extension of π-calculus, it inherits all its properties. This gives the possibility to manipulate some terms that are free within the π-calculus limitations. Manipulated terms will be called processors and will have at their disposal data streams they can use.

During the orchestration activation, the definition is transformed into an instance. Activation is made via a component that is one of the different tools defined in the meta-meta-model. These tools are defined as terms in the section dedicated to the definition of the system.

The definition of an orchestration considers the definition of a general context of the process as shown in Figure 9.This context allows the exchange of shared information between the various components of the orchestration. This set of shared variables is a part of the state context of the business agent at a given time. The result of the invocation of a route will depend on the current state of the agent because a previous invocation may have set a value on a shared variable, and thus influence the final result.

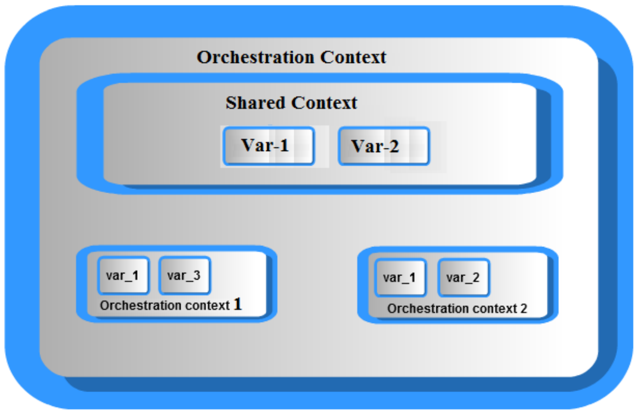


Figure 9. Shared context

The semantics of the agent is enhanced after loading the definition by the engine. The engine activates the orchestration routes and thus integrates the wrappers (Consumers and Providers). Then, the engine loads the context of the agent. Following this action, we end up with an active and ready-to-receive external invocations system state. However, it is important to make the distinction between the contexts of the agent corresponding to the internal information of the agent on one side and the state of the system that contains the context and the routes constituting the different agents on the system.

V. SYSTEM DEFINITION

Based on the definition (1), our system (9) defines a container running in parallel with the Repository.

$$System \overset{\text{def}}{=} (\nu\, repoGet\ repoPut)$$

$$Container(repoGet, repoPut) \mid Repository(repoGet, repoPut) \qquad (9)$$

The Repository (10) is a term that represents a composition for sharing the definition of agents. It can add an artifact containing the definition of an orchestration or retrieve the artifact using the URL that was used to add the artifact. The processing performed inside the Repository complies with the Maven [29] specifications. We will ignore the details of the inner workings in this paper.

$$\begin{aligned} Repository(repoGet, repoPut) &\overset{\text{def}}{=} \\ &\underline{repoGet}(paxuri_i, http).\tau \\ &.\overline{http}\langle bundle_i\rangle.Repository \\ &+ repoPut(paxuri_j, http) \\ &.http(bundle_j).\tau.Repository \end{aligned} \qquad (10)$$

The container (11) is the container application on which our services and our agents will be deployed. It allows loading definitions of orchestrations in its context. The container and the system have the same execution context.

A container can host any number of agents and services. Because each agent/service has a definition of its own, let's take the example of a system that contains one agent that performs an orchestration using a couple of services. The container allows the sharing of different channels to activate the definition of an agent in the engine.

Shared channels are associated with EIP. The definition of orchestration is transformed after activation in a set of Routes respecting an EIP sequence.

In order not to overload our definitions with a large number of parameters we will use the name "EIP" to represent all EIP names.

$$\begin{aligned} &Container(repoGet, repoPut) \overset{\text{def}}{=} \\ &(\nu\ EIP)\ (\ (\nu\ l, install, start, uninstall\ stop) \\ &(Runtime(l, install, uninstall\ ) \\ &\mid Engine(l, EIP, start, stop)) \\ &\mid (\nu\ uri_a,\ uri_b) \\ &\ (Agent(EIP) \mid ServiceA(uri_a) \mid ServiceB(uri_b))\ ) \end{aligned} \qquad (11)$$

Runtime (12) is designed to: manage the retrieval, activation and shutdown of various artifacts containing the definition of the agent as well as services. For this, it communicates with the Repository to recover the definition using the URL of the artifact. Once the artifact is recovered, it executes the definition to activate the engine.

$$\begin{aligned} Runtime(l, install, uninstall) &\overset{\text{def}}{=} (\nu\ \ http, r) \\ &(install(paxUrl, status) \\ &.repoPut(paxUrl, http).http(bundle_i) \\ &.MapPut(l, bundle_i, r).r(id).\overline{status}\langle id\rangle) \\ &+ (\nu\ \ http)(\ uninstall(id, status) \\ &.MapRem(id, r).\overline{status}\langle id\rangle) \end{aligned} \qquad (12)$$

The Engine (13) enables The Routes activation. Routes will be added to the system's context. The integration of context changes their status. The new status supports invocation of the active orchestration.

$$\begin{aligned} &Engine(EIP) \overset{\text{def}}{=} \ (\nu\ l) \\ &\quad Routes(l, EIP) \mid RouteActivator(l) \\ &RouteActivator(l) \overset{\text{def}}{=} \ (Processeur(l_i) \bullet)^{\|l\|} \\ &\mid (\nu\ \ http)(\ start(id, status).MapPut(id, l) \\ &.r(bundle_j).\overline{status}\langle id\rangle.RouteActivator(l) \\ &+ (\nu\ \ http)(\ stop(id, status).MapRem(id, r) \\ &.r(bundle_j).\overline{status}\langle id\rangle.RouteActivator(l) \end{aligned} \qquad (13)$$

The term Routes (14) is the basic element of the activation of an orchestration, as the term that uses the "emissions" on EIP channels. It is able to add to the system the ability to run the orchestration, then, transform this definition to a set of steps that are executed after the event fired.

$$Routes(l, EIP) \ \overset{\text{def}}{=} \ from(uri).Route(l, uri, EIP) \qquad (14)$$

The term Route (15), as its name suggests, allows you to link an entry to one or more outputs. Routing the term can manage a set of connections between both ends with a transform in the stream exchanged if needed.

$$\begin{aligned} &Route(l, uri, EIP) \overset{\text{def}}{=} \\ &\quad process(P).MapPut(l, Processeur\ (P)) \\ &\quad + from(uri).MapPut(l, Consumer(uri)) \\ &\quad\quad + to(uri).MapPut(l, Provider(uri)) \end{aligned} \qquad (15)$$

The first step is the transformation of a π-DSL definition to data structure representing an orchestration. This transformation is conducted by the term 'Routes' listening on the EIP channels. At each "emissions", the term Route manages the integration of a Route in the current orchestration. To do this, the term 'Routes' Delegates the treatment of integration PIEs to orchestration. Therefore, appealed to the term Route after each transmission on channel EIP 'from'.

The second level of transformation is the transformation of the structure representing a Route in a set of processes chained together and able to implement the semantics of the orchestration

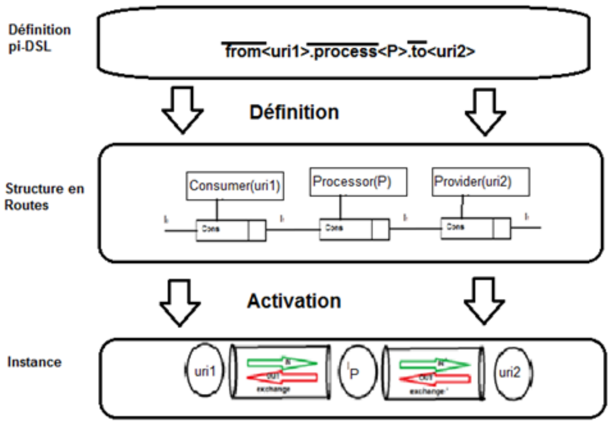


Figure 10. Activation of orchestration

This subdivision illustrated in Figure 10 allows us to keep control of an intermediate data structure, which may be modified to adapt it to the target platform. This transformation is at the heart of the migration mechanism that we will detail in a future paper

VI. CASE STUDIES

In order to illustrate our approach by case studies, we will take as an example the definition of an orchestration between two weather services and compare the values returned by called services.

We begin by defining our orchestration that will be as shown in Figure 11:

$$\overline{from}\langle uri_1\rangle.\overline{process}\langle P_1\rangle.\overline{to}\langle uri_2\rangle.\overline{process}\langle P_2\rangle.\overline{to}\langle uri_3\rangle$$

Figure 11. Generated Camel-DSL code

This definition is subject to an automatic transformation (as shown in Figure 7) of π-DSL part, against the terms {P1, P2} that represent the processor, which will be subject to manual transformation.

A mapping is defined between the pair {P1, P2} and there collocations in a π-DSL definition. The result will be in the form of Camel DSL code ready to be loaded and run on tools materialized from the meta-meta-model. Tools are generated in the form of a container, which uses Apache Felix [30] as a basis for implementing the definition of the container.

The second tool is the repository, which is an implementation standard Apache maven.

The third is the runtime that is included in the OSGi container (Felix) and provides a shell "Gogo" for interacting with the external.

Go back to our example of the definition of agent orchestration. The transformation from the π-DSL in code "Camel-DSL" leads to a deployable artifact on the container. The code is as shown in Figure 12.

```
import org.apache.camel.builder.RouteBuilder;
/**
 * A Camel Java DSL Orchestration
 */
public class OrchestrationRouteBuilder extends
RouteBuilder {

    public void configure() {

        from("nmr:uri1")
                .process(p1)
                    .to("nmr:uri2")
                .process(p2)
                    .to("nmr:uri3");
    }

}
```

Figure 12. Generated Camel-DSL code

Once deployed and activated, this route allows us to integrate the services present on the uri2 and 3 with the client that invoked the uri1.

The Camel engine will take control of the artifact deployed and ensure the interpretation of the Camel-DSL code. The engine will incorporate routes contained in the artifact to its execution context. The result will change the state of the system initially defined by the tools generate during the transition from meta-meta-meta-model to model.

The system is then enriched by the definition of the agent. Activation of this definition enhances the overall execution context.

VII. CONCLUSION

In this paper, we were able to develop an approach for generating a system dedicated to the orchestrations. Our approach is based on the MDA approach to obtain a dedicated orchestration and a set of tools constituting the execution context of the π-DSL orchestration

The formalism represented by the π-DSL language, defines an orchestration as a composition EIP. The orchestration is transformed into a camel-DSL and packaged as Maven artifact. The activation of the archetype load routes EIP composes orchestration.

We will discuss in a forthcoming paper on mobility in order to include in the definition of our system. We will prove by model checking [31] the mobility support of the system code.

We propose an extension of the semantics of our approach by adding a new dimension of freedom through the mobility aspect, which will be added to the semantics of an orchestration.

REFERENCES

[1] C. Peltz, “Web Services Orestrestration and Choreography,” Computer, vol. 36, no. 10, Oct. 2003, pp. 46-52

[2] BPMN. Bpmn - business process modeling notation. ‘http://www.bpmn.org/ retrieved: October, 2013

[3] C. Ibsen and J. Anstey, Camel in Action, Manning Publications, 2010

[4] C. Walls, R. Breidenbach, Spring in Action, 2nd Ed, Manning Publications, 2008

[5] M. Endrei et al., Patterns: service-oriented architecture and web services. IBM Corporation, International Technical Support Organization. 2004.

[6] G. Hohpe and B. Woolf, Enterprise Integration Patterns : Designing, Building, and Deploying Messaging Solutions . Addison-Wesley, Boston, 2004.

[7] D. Chappell, Enterprise Service Bus, O‟Reilly Media, Inc., Sebastopol, 2004.

[8] A.Charfi and M. Mezini, “Hybrid Web service composition: business processes meet business rules,” Proc. ICSOC ‟04, Proceed- ings of the 2nd international conference on Service oriented computing, ACM Press, New York, 2004, pp. 30–38.

[9] D. B. Lange and M. Oshima, “Seven good reasons for mobile agents,” Commun. ACM , vol. 42(3), 1999, pp. 88–89.

[10] C. Mahmoudi and F. Mourlin, “Adaptivity of Business Process,” Proc. ICONS 2013, The Eighth International

[11] OSGi Alliance. OSGi Service Platform Core Specification , release 4, version 4.2 ed. 2009 http://www. osgi.org retrieved: October, 2013.

[12] G. B. Laleci et al., “A Platform for Agent Behavior Design and Multi Agent Orchestration,” Agent-Oriented Software Engineering Workshop, the Third Inter- national Joint Conference on Autonomous Agents & Multi- Agent Systems, 2004, pp 205–220.

[13] BonitaSoft. Bonitasoft : open source business process management and workflow software. URL : http://www.bonitasoft.com/ Retrieved on January 25, 2013

[14] T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design ; Prentice-Hall, 2005.

[15] S. P. Fonseca, M. L. Griss, and R. Letsinger, “Agent behavior architectures a MAS framework comparison,” Proc. AAMAS, 2002, pp. 86–87.

[16] M. Viroli, E. Denti, and A. Ricci, “Engineering a BPEL orchestration engine as a multi-agent system,” Journal of Science of Computer Programming, vol 66, issue 3, 2007, pp. 226-245.

[17] A. Charfi and M. Mezini, “Aspect-oriented web service composition with AO4BPEL,” ECOWS, LNCS, vol.

[18] D. Jordan and J. Evdemon editors. Web services business process execution language version 2.0.http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.pdf retrieved: October, 2013

[19] F. Stan and A. Graesser, "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents." Intelligent agents III agent theories, architectures, and languages. Springer Berlin Heidelberg, 1997, pp. 21-35.

[20] R. Monson-Haefel and D. Chappell, Java Message Services. O’Reilly, 2001.

[21] R. Milner, The polyadic p-calculus: a tutorial. Technical Report ECS-LFCS-91-180, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, UK, October 1991. Also in Logic and Algebra of Specification, ed. F. L. Bauer, W. Brauer and H. Schwichtenberg, Springer-Verlag, 1993.

[22] R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes, Parts I and II . Volume 100 of Journal of Information and Computation , pages 1-40 and 41-77, 1992.

[23] D. Sangiorgi, “From -calculus to Higher-Order -calculus | and back,” Proc. TAPSOFT, LNCS 668 . Springer-Verlag, 1993.

[24] A. Kleppe, S. Warmer, and W. Bast, MDA Explained. The Model Driven Architecture: Practice and Promise, Addison- Wesley, April 2003.

[25] S. R. Judson, R. B. France, and D. L. Carver, Specifying Model Transformation at the Metamodel Level, Wisme 2003.

[26] M. Campione and K. Walrath, The Java Tutorial. Addision-Wesley, 2003.

[27] G. Benguria, X. Larrucea, B. Elvesæter, T. Neple, A. Beardsmore, and M. Friess, “A platform-independent model for service-oriented architectures,” Proc. I-ESA‟06, 2006.

[28] R. Z. Frantz, “A DSL for enterprise application integration,” International Journal of Computer Applications in Technology, vol. 33(4), 2008, pp. 257–263.

[29] Maven , In Apache Maven Project, http://maven.apache.org/ Retrieved on January 25, 2013

[30] Apache felix. http://felix.apache.org/site/index.html Retrieved on January 25, 2013.

[31] B. Bérard et al., “Systems and Software Verification,” Model-Chec king Techniques and Tools, Springer, 2001.

Relationships Between Risks in an IT Project Development Portfolio

Rob J. Kusters
Depts. Of Management Science
Open University
Heerlen, The Netherlands
Rob.kusters@ou.nl

Jort J. Postema
i-Dienst
H2O
Oldebroek, The Netherlands
jortjp@gmail.com

Jos J. M. Trienekens
IE&IS
Eindhoven University of Technology
Eindhoven, The Netherlands
j.j.m.trienekens@tue.nl

***Abstract*— More and more it is seen that IT (Information Technology) projects are managed as a whole as part of a IT project portfolio. As one of the arguments for doing so, risk management at the portfolio level was identified as one of the advantages that could benefit from this. This was based on the notion that risks are not independent from each other and that an understanding of relationships between risks should support portfolio management. Given this origin it is somewhat surprising that the notion of relationships between risks does not play a part in IT portfolio literature. This prompted this research project aimed at investigating the existence and relevance of risk relationships in practice. A series of interviews with experienced IT project portfolio managers confirms both the existence and relevance of the risk relationships providing a basis for further research.**

Keywords-portfolio project management; risk management; risk relationships

I. INTRODUCTION

IT projects are managed as a whole as part of a IT project portfolio. As a concept this was proposed as early as 1982 by McFarlan [17], who, as one of the arguments for doing so, identified risk management at the portfolio level as one of the advantages that could benefit from portfolio management. He based this on the notion that risks are not independent from each other and that an understanding of relationships between risks should support portfolio management.

The importance of using risk management at the portfolio level is evident [21]. Interactions between projects, in terms of shared scarce manpower and usage of project results in other projects, are unavoidable. Ignoring these will lead to more problems than taking them into account. Even for small organizations that means someone should monitor risks across projects. In larger organizations part-time or even dedicated portfolio managers are seen to take up this task.

Given the original argumentation by McFarlan, it is somewhat surprising that the notion of relationships between risks does not play any part in IT portfolio literature. This prompted this research project aimed at investigating the existence and relevance of risk relationships in practice. In this paper, we will first discuss the theoretical background of this study. Next, the research design used will be discussed, followed by the results of the study. Finally, a discussion of results and conclusions will be provided.

II. BACKGROUND

Risk, in the context of IT projects, can be defined as the possibility of an unfavorable outcome in terms of time, cost, or functionality of the final project deliverable [22]. There is an extensive body of literature on identifying risks for IT projects [19], and managing risk in IT projects [19][22][23][24][11].

Risk can also be discussed at the IT development portfolio level. Turner & Müller [21] give the following definition of such a portfolio “a portfolio of projects is an organization, (temporary or permanent) in which a group of projects are managed together to coordinate interfaces and prioritize resources between them and thereby reduce uncertainty”. De Reyck et al [4] state that: “the selection of projects to compose a portfolio should ensure that all areas of the organization’s strategy are properly addressed and that the portfolio is well balanced”. Risk is an important aspect of this balance [11] and therefor plays an important role when managing a portfolio.

This is also emphasized in the definition of portfolio management by McFarlan who states that within the context of a portfolio “assessing the risk of their projects, separately and in the aggregate, will help managers make more informed decisions and ensure more successful outcomes” [17]. He also states that risk analysis of individual projects should play a major part in selecting projects for such a portfolio since “risks in practical situations, of course, are not independent of each other; rather, they are closely related” [17]. McFarlan based his work on the still widely used financial portfolio theory as developed by Markowitz [16] who states: “Sometimes the addition of the risky security produces a more conservative portfolio than the addition of the conservative security. This illustrates a basic principle: the security which is risky or conservative, appropriate or inappropriate, for one portfolio may be the opposite for another. One must think of selecting a portfolio as a whole, not securities per se”.

Identifying portfolio risk can start by identifying all individual project risks and adding these to a single portfolio, see e.g., [3]. This approach can already provide significant insight. However, it misses the notion contended by

Markowitz and McFarlan that risks themselves can have relationships. If risks of individual projects can influence each other (across projects) these interactions should also play a role when making decisions of additions to a project portfolio.

When looking at literature for the management of a portfolio as a whole, attention has mainly be focused at interrelationships between projects. This relationship can be complementary, negative, or neutral [1][5]. Chien [1] identified four types of interrelationships among projects: outcome or technical, cost or resource-utilization, impact or benefit, and serial (present-value) interrelationships. Santhanam and Krypakis [18] identified three fairly similar types of interdependencies involving IT projects: resource, benefit, and technical. And in 2011 Kundisch & Meier [15] describe project interactions based in outcome or resource interaction. It is interesting to see that direct relationships between projects have received explicit attention, while relationships between risks receives no attention in this part of the portfolio literature.

Also, a wider search for literature aimed at identifying these relationships between risks in a IT project portfolio context yielded no results. In other fields the notion does exist. For example, Fan, Suo & Feng [7], when discussing the related area of IT outsourcing identify the existence of risk relationships. They state: “in some situations, the interrelationships among risk factors can induce the transmission effect from one risk to another”. In their research they elaborate further on this statement and identify eight relevant risks and their relationships. The notion of relationships between risks is also known in other disciplines. Examples are engineering [13], finance [6], and medical science [20][25].

Given this, it was found worthwhile to investigate the existence of relevant relationships between risks in an IT project portfolio setting.

III. Approach

The objective of this study is *to investigate if relationships exist in practice between risks of projects in an IT portfolio setting that are relevant at the portfolio level*. The notion of relevance has been added to the original question since slight interactions between phenomena can always exist, but from a management point of view these are only worth investigating if they have a significant effect on the management of the portfolio. The notion of ‘relationship between risks’ can now be further detailed. An obvious form exist when occurrence of risk X will impact the likelihood and/or the impact of risk Y. This can be termed a direct relationship between risks X and Y and can be interpreted as “if risk X occurs, this can influence the likelihood and/or the impact of risk Y”. A second type of relationship occurs when an external event can influence both the likelihood and/or the impact of risks X and Y (see Figure 1). In both cases the impact on likelihood and / or impact can be positive or negative, resulting in either a mitigating of aggravating effect on the portfolio level.

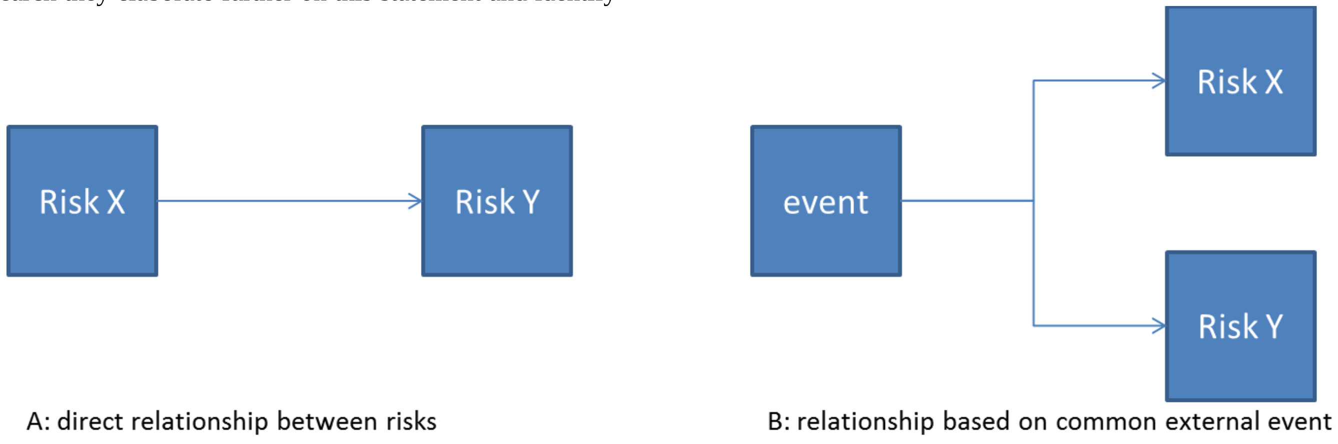

Risk X
Risk Y
A: direct relationship between risks
event
Risk X
Risk Y
B: relationship based on common external event

Figure 1. types of relationships.

Given the explorative nature of the research, and the fairly complex notions of ‘risk relationship’ involved it was decided to perform the research by interviewing a number of experts. This would provide the possibility of explaining the issues, seeing if these were understood and assessing the answers, also by asking additional questions if possible. These advantages of interaction, enabled by the interview format in our mind outweigh the more detailed and possibly more representative results that might be obtained from a survey.

For the interviews persons with relevant experience as IT project portfolio manager in a sizeable organization were sought. Two years or more of experience was required, since the expectance was, that this would provide the required relevant experience from which to answer our questions. A sample of five respondents from different organizations was aimed at. A larger number would of course have increased the number of identified relationships. However, given the objective: give a proof of existence of these risk relationships across projects, this was deemed to be sufficient. For the search use was made of relevant groups in Linkedin. In the end, five experienced IT-portfolio managers were found with the required profile who were willing to participate in the research (Table 1).

TABLE I. OVERVIEW OF RESPONDENTS

Respondent	Type of organization	Size
1	Energy provider	1.000-5.000
2	Government	10.000+
3	Insurance	1.000-5.000
4	University	5.000-10.000
5	Hospital	5.000-10.000

Respondents are supposed to provide concrete risk relationships they themselves have experienced. This is a fairly difficult questions to answer. To support their thought process it was decided to provide them with a short list of candidate risks to trigger them. To develop this list, an additional literature search was executed. The search was aimed at identifying 12 often used but dissimilar risks. The number of 12 was chosen as sufficiently small to be usable in an interview but also sufficiently large to be able to give material for discussion. For this, seven useful papers were selected:

- Risks that influence the risk profile of an IT project portfolio [17].
- A structured overview of risks: [11].
- Sources that can originate common risks: [19].
- Twelve dominant risks [12].
- A top 10 of software risks [9].
- A number of risks derived from failed projects [2].
- A recent publication containing critical risks [8].

The selection process of these papers took into account a number of quality criteria: an assessment of the methodology used and the number of times the paper was referenced.

An overview of all risks identified in these papers was developed. Overlaps between papers were identified and the risks were sorted according to frequency of occurrence in the papers. This resulted in the following list (with between brackets the number of papers in which the risk is mentioned):

- incorrect or misunderstood requirements (6)
- insufficient project planning (6)
- lack of non-IT human and/or financial resources (6)
- inexperienced IS project team (5)
- unclear project scope (5)
- insufficient project approach (5)
- lack of end user participation (4)
- changes in team composition (4)
- changes in project scope and / or requirements (4)
- lack of man power (IS related) (4)
- unfamiliarity with hardware in the project team (4)
- unfamiliarity with software in the project team (4)

In order to achieve results of sufficient quality a semi-structured interview set-up was developed. The interview started with a question regarding the work experience of the respondent in order to confirm their level of experience. This was followed by an explanation of the issues involved and the notion of risk relationship types (Figure 1). The objective was to explain the objective of the interview and the concepts involved. Part of this was a check on comprehension of these concepts, preferably by having the respondent explaining them in their own words.

This was followed by the key component of the interview: a discussion regarding possible risk relationships. To focus this discussion as a visual aid a (half) matrix was provided in which the risks identified were set off against each other. Also, a more detailed version of figure 1 was included as a memory aid.

Using the resulting matrix, respondents were prompted to identify relationships (direct and based on a common external event) between risks and to provide concrete examples of occurrences of these relationships which they personally encountered. The examples were required to ensure that only actually occurring risk relationships were identified and not just theoretical / hypothetical possibilities. No completeness in the discussion of all 66 possible combinations of risk was striven for. This would have been pointless in the limited time available for such an interview. Respondents could add risks on top of the twelve identified if this helped them in identifying additional risk relationships. These new risks were added to the risk matrix to be available for subsequent interviews. The basic question put to the respondents here was: do you have a specific relationship between risks from this matrix in mind which you want t discuss?

After this part of the interview, results from previous interviews were presented. Respondents could indicate if they agreed with them in principle, providing a face value validation of previous results.

The setup of the interview was tested beforehand with a test subject, who was not an active IT project portfolio manager but did have some experience with portfolio management. No changes were made to the set-up as a result of this test.

All interviews were recorded. The recording were transcribed and then analyzed. The analysis was aimed at identifying actual risk relationships discussed and the examples provided by the respondents. In recording these results, as much as possible the original statements made by the respondents were used. The results were send back to the participants for approval. Based on their feedback, some minor changes were made in the results.

IV. RESULTS

The interviews were carried out over a period of five weeks, allowing for sufficient time between interviews to have the results of a previous interview ready for the next. Of each interview an extended abstract was made, based on an audio recording. This abstract was sent back for confirmation to the respondents, who could make corrections.

All respondents have the required two years of IT portfolio management experience, ranging up to 10 year. The organizations involved are sizeable, indicating that the respondents have to deal with a significant IT-portfolio. Respondent 4 is also active as a consultant specialized in portfolio implementation and director / owner of a company specialized in portfolio management (> 20 employees). This indicates that a sufficient basis exists to accept the expertise of the respondents.

During the interviews, all respondents indicated that after some discussion they understood the concepts of risk relationship and the associate types of direct relationships and those based on a common external event. This, then provided a solid basis for the further interviews.

In the next step, all respondents were able to identify (direct and based on a common external event) risk relationships. They also were able to support this by providing concrete examples. As mentioned in the foregoing respondents were allowed to add risks if required for their discussion. All in all 5 additional risks were added to the matrix:

- Change in planning
- Benefits not achievable
- Portfolio out of control
- Common resource usage across projects
- Safety or security endangered

All-in-all 15 relationships were found, of which 7 based on a common external event and 8 direct. Table 2 gives an overview of the portfolio risk relationships found. The first seven lines of the table contain situations where the relationship is based on a common external event (situation B in figure 1). The remaining eight lines contain situation where a direct relationship between risks across projects exists (situation A in figure 1).

V. Discussion and conclusions

This discussion will look at the validity and reliability of the results, and their degree of completeness. It will end with a discussion of the added value of this notion, set off against approaches already in use.

Let us first look at the validity and reliability of the results. In this project five 2-hour semi-structured interviews were conducted with experienced IT project portfolio managers from fairly large organizations. These respondents all understood and recognized the phenomenon. Together they succeeded in identifying and validating fifteen risk relationships, of which eight direct and seven based on a common external event. All fifteen risk relationships were supported by concrete examples, based on their own experience. In consecutive interviews respondents were asked to confirm the existence of the earlier identified relationships. In interviews 2, 3 and 5 this was done. In interview 4 this proved not to be possible due to time constraints since discussion in the first part of the interview took too long. Interview 5 focused only on the validation of the previous results. In total this provided 27 options to confirm or deny a risk relationship. In 26 of these, existence of the relationship was confirmed, providing an additional face value support for its existence and relevance. In one case a relationship was accepted by one consecutive interviewee and denied by another. This is the relationship mentioned in the seventh row of the table in Table 2. Together this provides strong evidence of the existence of the phenomenon and the relevance of the relationships found. Together it can be concluded that the results are valid and reliable.

As mentioned above, the research was explorative and not aimed at achieving any degree of completeness. An indication of the degree of completeness achieved can be judged from the overlap between the relationships identified by the individual respondents. This is possible, since results from previous interviews were not shown until at the end of the interview. Of the fifteen relationships identified only two were identified more than once. Each was identified twice in different interviews. That means that four independent drawings (interviews) from a population of risk relationships of unknown size resulted in only two doubles. This would indicate that the results are far from complete and (many) other risk relationships are still to be identified.

When looking at the relevance of the results it is required to compare them with the approaches currently being used to see if any added value can be identified. In the background study two current approaches are identified. A first approach identified is adding individual risks to a portfolio risk profile see e.g., [3]. It is obvious that such an approach is likely to miss the additional insight in risk and benefit offered by the notion of risk relationship proposed here. The notion of risk relationship can be considered as a straight add-on to this approach. A second approach looks at describing project interactions e.g., in outcome or resource interaction (e.g., [15]. Such an approach is unlike to identify the common external events that are at the basis of some of the risk relationships identified in this study. The direct risk relationships could also be identified when looking at direct interactions between projects. However, the more detailed and forward looking approach enabled by the view on risk relationships is probably a useful addition to this approach.

VI. Conclusions and future work

Following this discussion, we conclude that the notion of risk relationship in the context of IT project portfolio management is a useful addition to the current state of the art and merits further research. Such relationships do appear to exist and are unlikely to be fully captured by existing approaches. This holds especially for the notion of external events impacting several risks across projects. Further research could be directed at providing a more complete overview of relationships as depicted in figure 1 and table 2. Extending the approach used in this research seems not feasible. There are not that many experienced project portfolio managers around willing to invest the large amount of time required for the required structural analysis.

Given that a structured literature review would yield a list of risk factor far larger than the one used in this research such a set of interviews would need to discuss hundreds of risk combinations, each again in combination with dozens of possible external events, leading to thousands of items to be analyzed.

A more feasible approach might be found in the analysis risk documentation, e.g., as captured in risk repositories. That would also be more directed (looking at actual occurrences) while not trying to cover an extreme number of combinations of which probably only a limited number yield results.

TABLE II. OVERVIEW OF PROJECT RISK RELATIONSHIPS

External event Z	Risk X	Risk Y	Example
Change in organization (culture)	incorrect or misunderstood requirements	changes in team composition	This type of change can lead to outflow of current staff. This will then influence both the understanding of requirements by new staff and will immediately impact team composition, with the entailing loss of common project understanding.
Change in organization (culture)	lack of man power (IS related)	benefits not achievable	The change caused a difference in usage of the document management system which impacted the effectiveness of running projects. It also caused outflow of current staff.
Market competition stronger	changes in team composition	change in planning	Competitive pressure caused moving deadlines forward. Due to unreasonable pressure projects got out of hand. This also caused outflow of staff.
Change in labor market	lack of (non-IT) human and / or financial resources	lack of man power (IS related)	Staff with specific competences left for higher wages. This caused a lack of these competences within the organization. Similarly, hiring temporary replacement staff became too expensive.
Change in (marketing) policy	changes in project scope and / or requirements	change in planning	The change resulted in new projects, resulting in delay and higher risk because of the delay for other projects. Also, other projects were required to change their scope to fit in with the new projects.
New legal requirement	changes in project scope and / or requirements	safety (or security) endangered	Decentralization of youth care to civic communities impacted the scope of projects for existing suppliers. Also, because of this decentralization, security risks increased.
Downsizing due to external circumstances	inexperienced IS project team & lack of (non-IT) human resources	lack of man power (IS related)	In a downsize situation the best staff had a tendency to leave (because they can). This resulted in lack of manpower and experience.
	changes in project scope and / or requirements	changes in project scope and / or requirements	When a project was faced with a change of scope, this directly impacted the scope an output related project.
	lack of financial resources	changes in project scope and / or requirements	When a project consumed too much resources, this directly impacted the availability of the (remaining) resources for the other / later projects.
	changes in project scope and / or requirements	benefits not achievable	When a project adjusted its scope, an output related project was unable to achieve its objectives.
	lack of man power (IS related)	lack of financial resources	Staff works on several projects. A specific project is put on hold. As a consequence, the capacity that became available was absorbed by the other projects, increasing their costs.
	change in planning	lack of man power (IS related)	A project required specific and scarce capabilities. When the project ran late, this capability was not available for other projects, who all ran late as well.
	portfolio out of control	insufficient project planning	A program with many dependencies between projects ran out of control. The result was that planning of these projects could not be maintained.
	common resource usage across projects	insufficient project approach	An organizations used configuration management tools of insufficient quality. This impacted the entire portfolio.
	lack of financial resources	lack of financial resources	A specific project had lack of funding. Portfolio management challenged all other projects to work more efficient in order to release the required funding.

Another field of research is the notion of external event. It could be envisaged to do further research into the type of events that could impact project risk and thus provide a reference that can be used by portfolio managers to support their work. Finally, it could be worthwhile to investigate the strength of the relationships identified and the likelihood of occurrence.

REFERENCES

[1] Chien, C.-F. (2002). Portfolio-evaluation framework for selecting R&D projects. R&D Management, 32(4), 359-368.

[2] Chua, A. (2009). Exhuming it projects from their graves - an analysis of eight failure cases and their risk factors. Journal of Computer Information Systems / spring, 31-39.

[3] De Giorgi, Enrico G., A Note on Portfolio Selection under Various Risk Measures (August 2002). Available at <http://dx.doi.org/10.2139/ssrn.762104> 21-08-2013.

[4] de Reyck, B., Grushka-Cockayne, Y., Lockett, M., Calderini, S., Moura, A., & Sloper, A. (2005). The impact of project portfolio management on. International Journal of Project Management, 524-537.

[5] Devinney, T. M., & Stewart, D. W. (1988). Rethinking the product portfolio: A generalized investment. Management Science, 34(9), 1080–1096.

[6] Dhaene, J., & Denuit, M. (1999). The safest dependence structure among risks. Leuven, België: Katholieke Universiteit Leuven.

[7] Fan, Z., Suo, W., & Feng, B. (2012). Identifying risk factors of IT outsourcing using interdependent information: An extended DEMATEL method. Expert Systems with Applications 39, 3832-3840.

[8] Hajeer, S. (2012). Critical risk factors for IS projects IS project between sink and swim. International Journal of Communication Engineering and Technology / June, Vol 2, issue 6, 1270-1279.

[9] Han, W., & Huang, S. (2007). An empirical analysis of risk components and performance on software projects. The Journal of Systems and Software 80, 42-50.

[10] Heemstra, F.J. and Kusters, R.J. (1996). Dealing with risk: a practical approach, Journal of Information Technology, vol. 11, pp. 333-346.

[11] Heemstra, F.J., and Kusters, R.J. (2004). Defining ICT Proposals. Journal of Enterprise Information Management, Special Issue on IS Evaluation, vol. 17, no. 4, pp. 258-268.

[12] Kappelman, L., McKeeman, R., & Zhang, L. (2006). Early warning signs of IT project failure: the dominant dozen. Information Systems Management / fall, 31-36.

[13] Karningsih, P., Kayis, B., & Kara, S. (2007). Risk Identification in Global Manufacturing Supply Chain. International Seminar on Industrial Engineering and Management (pp. 8-15). Jakarta: ProdEff Technology.

[14] Kumar, Ram , Haya Ajjan, and Yuan Niu (2008). Information technology Portfolio Management: literature review, framework, and research issues, Information Resources Management Journal, Volume 21, Issue 3.

[15] Kundisch, D., & Meier, C. (2011). IT/IS Project Portfolio Selection in the Presence of Project Interactions – Review and Synthesis of the Literature. Proceedings of the 10th International Conference on Wirtschaftsinformatik - Volume 1 (pp. 477 - 486). Zurich, Switzerland: Lulu.com.

[16] Markowitz, H. (1959). Portfolio Selection - efficient diversification of investments. In H. Markowitz, Portfolio Selection - efficient diversification of investments. New York: John Wiley and Sons, inc.

[17] McFarlan, F. (1981). Portfolio approach to information systems. Harvard Business Review, 142-150.

[18] Santhanam, R., & Kyparisis, J. (1996). A decision model for interdependent information system project selection. European Journal of Operational Research, 89(2), 380–399.

[19] Schmidt, R., Lyytinen, K., Keil, M., & Cule, P. (2002). Perceptions of IT project risk: A Delphi study. Information Systems Journal, 12(2), 103–119.

[20] Sorrentino, G., Migliaccio, R., & Bonavita, V. (2008). Treatment of Vascular Dementia: The Route of Prevention. European Neurology, 217-223.

[21] Turner, J., & Müller, R. (2003). On the nature of the project as a temporary organization. International Journal of Project Management 21, 1-8.

[22] Wallace, L., & Keil, M. (2004). Software project risks and their impact on outcomes. Communications of the ACM, 47(4), 68–73.

[23] Westerman, G. (2005). What makes and IT risk management process effective. MIT Sloan School of Management Center for Information Systems Research, 5(3B), 1–3.

[24] Westerman, G., & Walpole, R. (2005). Working article: PFPC: Building an IT risk management competency. CISR, 1–13.

[25] Wilson, P., Abbot, R., & Castelli, W. (1988). Arteriosclerosis thrombosis and vascular biology. Journal of the American Heart Association, 737-741.

A Proposal of Requirements Specification Process for Adaptive Systems Based on Fuzzy Logic and NFR-Framework

João DionisioParaiba
FACEN – Faculty of Exact and Natural Science
Methodist University of Piracicaba (UNIMEP)
Piracicaba, SP – Brazil
jdparaiba@gmail.com

Luiz Eduardo G. Martins
UNIFESP – Science and Technology Department
São José dos Campos, SP – Brazil
legmartins@unifesp.br

***Abstract*— Fuzzy Logic is a concept that deals with ambiguities, uncertainties and vague information on the solution of problems. NFR-Framework deals with the non-functional requirements which also are, very often, vaguely and full of uncertainties. In this paper, we use these concepts to propose a process for requirements specification of adaptive systems, called PERSA - Portuguese acronym to "Processo de Especificação de Requisitos para Sistemas Adaptativos". Adaptive systems consist of functional and non-functional requirements, which hold the capacity to modify themselves during the runtime with little or no human intervention at all. However, despite being a very discussed topic in Requirements Engineering (RE) community, it still lacks tools and techniques to standardize its modeling. The proposed process is instantiated in a case study which is discussed along this paper.**

Keywords-Adaptive Systems, Adaptive Requirements, Requirements Specification, Fuzzy Logic, NFR-Framework.

I. INTRODUCTION

The continuous evolution of software systems, the increase in complexity and the integration of technology, among other factors, lead the Requirements Engineering (RE) community to seek inspiration in some related areas (Robotics, Control Theory and Biology), in the attempt of finding innovative approach to the building and management of software systems. Therefore, adaptive systems are able to set their behavior at runtime as an answer to the environment and to the system itself, making it a very discussed theme in the RE community [1]. Adaptive systems have grown in importance with the increasing complexity of software systems and the need of such systems to be versatile, flexible, reliable, robust, recoverable, customizable, self-sustained and optimized, since they deal with these characteristics and with uncertain contexts which are often not discussed in the specification process, then requiring the system to adapt to unexpected changes. Adaptive system is a new frontier for RE community and industry setting.

The most common use of adaptive systems is in the previously mentioned areas of robotic and control theory, which demand dynamic readings of the context and immediate response to the system with as little human intervention as possible. The development of these systems has been significantly more challenging than the traditional model due to the need of mechanisms to automate and simplify the adaptation and modification of software after its installation [2]. Despite this, software engineers have focused their research on development of new technologies to manage the progressive complexity of software systems. The RE community and industry practitioners still lack templates and patterns to help and minimize the cost of developing such systems. It is noted in these circumstances the immense difficulty of specifying requirements for adaptive systems without previously defined and satisfactorily utilized pattern or tool.

Adaptive systems, as the name suggests, need to adapt to new context, but contextual uncertainties make it difficult to create, validate and manage the requirements. These systems are able to adjust their behavior at runtime as a response to the new reading of the context where the system is inserted [3]. However, despite being a very discussed topic in RE community, it still lacks tools and techniques to standardize its modeling.

RE technique and tools are satisfactory when the context is well known or evolves slowly. However, there is a need of mechanisms which automate and simplify the adaptation and modification of the system to operate in volatile contexts. The purpose of this research is to propose a specification process to adaptive systems focusing the definition of requirements that demand system adaptation. Such proposal is based on using Fuzzy Logic [4] and NFR-Framework [5].

Efforts to develop this research included a literature review on adaptive systems, requirements engineering, Fuzzy Logic and NFR-Framework. Such review aimed providing a theoretical basis for the definition of the object of the research that this study intends to produce. The activities began with a study about adaptive systems in general and about the works already produced by the RE community concerned to these systems. Papers and articles that dealt with these techniques and tool for specification and modeling of adaptive systems requirements were searched.

It was observed in the literature review that to manipulate requirements that go through changes at runtime, studies with Fuzzy Set Theory could be helpful. Several articles related to the context of adaptive systems with set theory were researched. It was found that in the context of adaptive systems, it would be viable to approach Fuzzy Logic context [6][7][8][9], due to its use in problems involving fuzzy contexts.

Next, a model able to cover this complex context of requirements for adaptive systems was sought, opting for this NFR-Framework, which deals with uncertainties through the concepts of softgoals and represents them satisfactorily by means of SIG diagrams. The next step was to map the contexts explored, making a relationship among the three areas studied: adaptive systems, Fuzzy Logic, and NFR-Framework. To finish the relationship identified in the mapping, it was realized that the concept of requirements for adaptive systems should be better characterized. After this characterization, later called adaptive requirements, it was noted the need of creating a conceptual model. For the representation of such a model, a class diagram (from UML) was adopted, which shaped the main concepts involved, based on a previously done array of mapping.

The rest of this paper is organized as follows: an overview about adaptive systems and requirements for such systems are presented in section II; a proposal of requirements specification process for adaptive systems is presented in section III; a case study using the suggested proposal is reported in section IV; and conclusions and further works are presented in section V.

II. ADAPTIVE SYSTEMS

Adaptive Systems are those that can be modified at runtime, due to changes in the system, in requirements or in the environment where they are implanted [3], depending upon various aspects, such as particular properties of a system, users requirements and characteristics of the environment.

According to Cheng [1], the simultaneous boom of information, the integration of technology and the continuous evolution of systems based on ultra large-scale software require new and innovative approach to building, implementing and managing software systems. To support this evolution, systems must become versatile, flexible, adapted to the three aspects mentioned above. To achieve this, the adaptive systems have become a topic of great interest in current researches in the Software Engineering Community [10].

There are requirements that are sensitive to the context in which the system will be implanted. Where the context is well known and static or evolutes slowly, the existing RE techniques can perform a good job. What is noticeable is that, increasingly, development projects are being challenged to build systems able to operate in volatile context, so that they are not totally previously understood [11][2].

Such systems must have the ability to dynamically adapt to new environmental context, but the contextual uncertainty that requires this adaptive potential hinders the elaboration, validation and management of its requirements and can be varied according to environmental requirements. The unexpected contexts may even lead to new requirements [3][12][13].

A. *Requirements for Adaptive Systems*

A conventional requirement (functional or non-functional) can be defined as a declaration of a service or constraint of a system being developed. It can also be simply defined as “something the client needs”. However, from the developer point of view, a requirement can also be defined as “something that needs to be developed”.

Developing adaptive systems demands making explicit the alternatives to achieve the goals, i.e., the variability in which and how it can be enhanced and the variability where and when, due to the operational environment.

This leads to the definition of requirements that are not only functional or non-functional, but also the specification of monitoring that takes under consideration the variability on an operational context, evaluation criteria and the behavior of alternative software being adopted by the software system at runtime to ensure the achievement of the user`s goals [14]. Requirements for adaptive systems are those that include the notion of variability associated to any functionality or a system quality constraint. Software requirements are generally characterized over the functional and non-functional classification. During the elicitation, the analyst first gives attention to the characterization of the stakeholders’ needs, which can be obtained through interviews or documents in a natural language. Requirements for adaptive systems reflect the uncertainties about the conditions at runtime due to the variability in the operational context and in the user`s necessities. In summary, adaptive systems are based on requirements that specify the necessity to modify the system behavior at runtime. Hereafter, requirements for adaptive systems with this characteristic are called adaptive requirements (AR).

III. PERSA: REQUIREMENTS SPECIFICATION PROCESS FOR ADAPTIVE SYSTEMS

This section presents the basic lines of the approach to the Requirements Specification Process for Adaptive Systems (PERSA – Portuguese acronym to *Processo de Especificação de Requisitos para Sistemas Adaptativos*). The process aims to aid the adaptive requirements specification activities through a well defined set of activities. Fuzzy Set Theory allows treating factors, such as ambiguity and uncertainty. Thus, the Fuzzy Sets, Fuzzy Logic and Fuzzy Reasoning provide the basis to generate the techniques to solve problems with a large applicability, especially in the control and decision making areas. In this work, the universe of fuzzy concepts formed by Fuzzy Set Theory, Fuzzy Logic and Fuzzy Reasoning will be mentioned as Fuzzy Logic. The NFR-Framework*,* which allows developers to work with the non-functional requirements, systematically expressing and using them to guide the development process of software systems. The NFR-Framework has the softgoals as main component, which have a subjective nature.

PERSA process used Fuzzy Logic concepts as a basis for its development since they treat factors, such as ambiguity, uncertainty, and vague information in the solution of problems, enabling handling adaptive requirements, as well as NFR-Framework concepts, which has the definition of softgoals, fully compliant to the modeling of uncertain requirements, providing notation and semantics for the construction of SIG diagrams, which will be used as a graphic representation for adaptive requirements. The Fuzzy Logic concepts applied to PERSA process were entirely used and there was no expansion or alteration. The NFR-Framework concepts, also entirely used, will shape the process when building the SIG diagram and the adaptive requirements and not only the functional and non-functional requirements.

Thus, this work has begun with the challenge of creating an approach for adaptive system based on requirements (functional and non-functional), which may undergo variations during their lifespan. Requirements suffering variability, changes or extensions at runtime are classified as adaptive. Process aims to specify requirements for adaptive systems handling them with the Fuzzy Logic concepts and shaping them with NFR-Framework concepts.

The initial stage of requirements specification deals with the definition of global aspects of the project, determining items such as: project purpose, project scope and functional areas involved; goals to be achieved; technical and business assumptions that affects the project; critical factors for the success, among others. It is important to remember the necessity of being previously defined. This way, the activity of collecting functional and non-functional requirements must be performed in a conventional manner. The analyst may use any modeling technique available in the RE community. The PERSA Process begins its life cycle right after the stage of requirements collecting.

A. Conceptualization

As mentioned above, adaptive requirements (AR) are those which include the notion of variability associated with any functionality or with any quality constraint of the system [15]. The first step in the creation of PERSA process consisted of the attributes identification for each concept related to an adaptive system concept:

User`s goals: what the software must meet. The user`s goals must be achieved.

Environment Variability: the environmental context where the software is implanted can change.

Alternative Behavior: according to a new reading of the environmental behavior, the behavior of the software may change.

Mutant Variables: are those which do not offer a clear definition of all values they may take. For example, the variable “fire intensity” may have values like high, middle or low.

Evaluation Criteria: an analysis of the software is performed after a change to check it is still meets the user`s goals satisfactorily.

Below the list of attributes of Fuzzy logic:

Linguistic Variables: have values with names of Fuzzy Sets. They can be put in a specific language, from primary terms, logic connectives, modifiers or delimiters.

Membership Functions: each Fuzzy Set is characterized by the membership function.

Fuzzification Interface: identifies the input variables values, which characterize the state of the system which normalizes it in a universe of standardized speech.

Inference Rules: represent the model of the system to be controlled. They characterize the goals and the control strategy used by specialists.

Defuzzification Interface: consists in obtaining a single discrete value usable in a concrete action of controlling the real world from the obtained fuzzy output values.

The list of NFR-Framework attributes completes the group of concepts in which PERSA is based on:

Softgoals: represent and aid developers to work on non-functional requirements (NFR).

SIG Diagram: the representation and use of *NFR-Framework* are made through SIG Diagrams.

Evaluation: determines the degree of satisfaction of the *softgoal* in its dependency relation with others.

Contribution: type of positive or negative collaboration to achieve the goals.

Interdependencies: are inter-relations between the softgoals refinements aiming the satisfaction of the related softgoals.

Catalogues: store the acquired knowledge structuring and enabling the reuse.

B. PERSA Process Activities

As previously reported, PERSA process starts right after the requirements survey ends. The PERSA process activities concerned with the creation of fuzzy rules was based on Mamdani method, which is a well known method to specify fuzzy rules. The PERSA process activities were organized in three main phases:

1st phase: Analysis of the Requirements List.

2nd phase: Fuzzy Modeling:

1st Stage: Create Linguistic Variables.

2nd Stage: Create Fuzzy Sets.

3rd Stage: Add values to the Fuzzy Sets.

4th Stage: *Fuzzification* Process:

1. Charge Input Values;
2. Choose Membership Function;
3. Perform Calculations according to Membership Functions;
4. Assemble *Fuzzification* Matrix.

5th Stage: Assemble Inference Rules:

1. Use *Fuzzification* Matrix;

2. Seek Specialist in Business Rule;
3. Choose Mamdani Method;
4. Build Knowledge Base according to Mamdani Method;
5. Interview Specialist;
6. Add data to the Knowledge Base;
7. Calculate Function MINIMUM;
8. Generate Graphic of Inference Rules;
9. Calculate Function MAXIMUM;
10. Generate Knowledge Base Graphic.

6th Stage: *Defuzzification* Process*:*

1. Use Knowledge Base Graphic;
2. Choose D*efuzzification* Method;
3. Use points from the Graphic of Knowledge Base;
4. Make Calculations.

3rd Phase: NFR Modeling

1st Stage: Specify Goals.

2nd Stage: Name NFR *Softgoals*.

3rd Stage: Generate SIG Diagram:

1. Create NFR *Softgoals*;
2. Decompose *Softgoals;*
3. Verify Operationalization;
4. Verify Decomposition;
5. Verify Correlation;
6. Select Operationalizations.

The input to PERSA process comes from the requirements elicitation performed in a conventional way. The elicited requirements are analyzed with the intention to find those that present variations during the adaptive system life cycle. That is the first phase of the process. For each requirement that presents meaningful variability, the 2nd and 3rd phases of PERSA process must be performed. At the 2nd phase a fuzzy model is created following the steps listed before. At the 3rd phase a NFR model is created, associating linguistic variables and fuzzy sets to the softgoals. The fuzzy model and NFR model complement each other, helping requirements engineers to better understand the adaptive requirements.

IV. Case Study

In this section a case study is presented, in which PERSA process was used integrally, aiming to specify adaptive requirements in the analyzed problems. The case study, called "cook`s problem", consisted of the specification of an automate system to prepare steaks, requiring an adaptive system related to the different types of meat, which are prepared according to the customers` order being rare, medium or well-done.

A. Cook`s Problem

As recommended by PERSA process, the input variables, the output and their respective fuzzy sets were initially defined, as showed in Tables I and II. In Figure 1, the graphics with the values of fuzzy sets of the variables "Time" are presented. The horizontal axis represents membership degrees and vertical axis represents the fuzzy sets thresholds.

TABLE I. Values Range of the Input Fuzzy Sets

	Reddish			Pinkish			Brown		
	Beginning	Average	End	Beginning	Average	End	Beginning	Average	End
Tone	155	205	235	95	125	175	55	75	135
Time	0.5	1	2	1	2	3	2.5	3	4.5
	Short			Average			long		
	Beginning	Average	End	Beginning	Average	End	Beginning	Average	End

TABLE II. Values Range of the Output Fuzzy Sets

2.	Output Variables														
	Rare			Medium			Well Done			Raw			Burnt		
	Beginning	Average	End	Beginning	Average	End	Beginning	Average	End	Beginning	Average	End	Beginning	Average	End
State	1	2.5	4	3	4.5	7	6	8.5	10	-5	-2	1	10	12	15

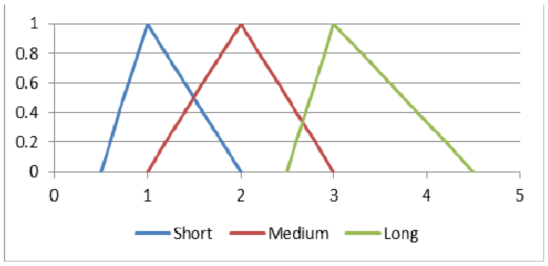


Figure 1. Input Variable "Time" with the values of Fuzzy Set

According to the fuzzy sets, the membership function triangular was chosen. In this case study, the system was fed with the values 98 for the input variable Tone and 1.9 for the input variable Time.

TABLE III. Fuzzification Matrix

Input Variable	**Input Value**	**Fuzzy Sets**		
Time	1.9	*μShort*	*μMedium*	*μLong*
		0.10	0.90	0.00
Tone	98	*μReddish*	*μPink*	*μBrown*
		0.00	0.10	0.62

The fuzzification matrix was made from the result of the Membership Function *Triangular*, according to Table III. With this done, it moved to the fifth stage of the second phase of PERSA process. At this point, the process requires a specialist to assist the definition of the system inference rules.

This fifth stage of the second phase may be considered essential since it contains the main difference between the adaptive and the conventional system. Here, the table Knowledge Base is constructed, based on Mamdani method, when the specialist determines the results of each combination among the input variables. In the cook`s problem case study, according to Table IV, it may be noted that the specialist`s answers are in the last column. For example, *If Short Time and Reddish Tone, then state of the steak = Raw*.

TABLE IV. KNOWLEDGE BASE MATRIX – COMPLETE

Reg.	Time		Tone		State	Minim.
	Fuzzy	Pert.	*Fuzzy Set*	Pert.		
01	*µShort*	*0.10*	*µReddish*	*0.00*	Raw	**0.00**
02	*µShort*	*0.10*	*µPink*	*0.10*	Rare	**0.10**
03	*µShort*	*0.10*	*µBrown*	*0.62*	Medium	**0.10**
04	*µAverage*	*0.90*	*µReddish*	*0.00*	Rare	**0.00**
05	*µAverage*	*0.90*	*µPink*	*0.10*	Medium	**0.10**
06	*µAverage*	*0.90*	*µBrown*	*0.62*	Well Done	**0.62**
07	*µLong*	*0.00*	*µReddish*	*0.00*	Medium	**0.00**
08	*µLong*	*0.00*	*µPink*	*0.10*	Well Done	**0.00**
09	*µLong*	*0.00*	*µBrown*	*0.62*	Burnt	**0.00**

To each rule created by the Mamdani Method and described in Table II, the function Minimum must be calculated and the graphics must be generated, which are the basis to create the Knowledge Base Graphic, illustrated in Figure 2. Through this, the Centroid is calculated and the mathematical data are transformed in numbers from the real world. In the case study, the inputs inform that the steak contains 56% (fifty six percent) of characteristics in the Fuzzy Set “Well Done” and thus, the fuzzification process is finished in PERSA Process.

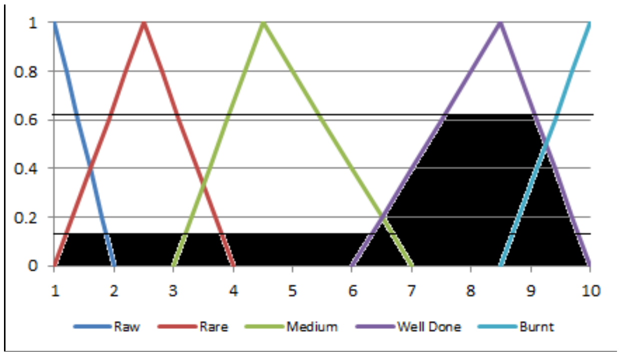


Figure 2. Knowledge Base Graphic

TABLE V. DEFUZZIFICATION METHOD ADOPTED IN THE CASE STUDY - CENTROID CALCULATION

Score in the Graphic	1.5	2.5	3.5	4.5	5.5	6.5	7.5	8.5	9.5
Rare	0.1	0.1	0.1						
Medium				0.1	0.1	0.1			
Well Done						0.2	0.6	0.7	0.7
MAXIMUM	0.1	0.1	0.1	0.1	0.1	0.2	0.6	0.7	0.7
*Score * MAXIMUM*	0.15	0.25	0.35	0.45	0.55	1.3	4.5	5.95	6.65
SUM (MAX)	**2.7**								
SUM (Score * MAX)	**20.15**								
RESULTADO	**7.462962963**								

The result of defuzzification method presented 7.46 in the output variable, showed in Table III. This means that with the inputs in the system (time = 1.9 min. and tone = 98), this Steak contains 56% (fifty six percent) of characteristics inside the Fuzzy Set “Well Done” and 0% (zero percent) membership in the other sets. Then, it can be said that the steak is “well done”.

The last phase of the PERSA Process, named NFR Modeling, generates SIG diagrams: to each input variable a NFR softgoal is created, as illustrated in Figure 3. It may be noted that the main difference between modeling in a conventional system and an adaptive one, through NFR Modeling is in Figure 3, exactly in the “Verify Inference Rules” softgoal. To meet this, three conditions must be met:

- The “Monitoring Color” softgoal must be Pink;
- The “Monitoring Time” softgoal must be Short or Average Time;

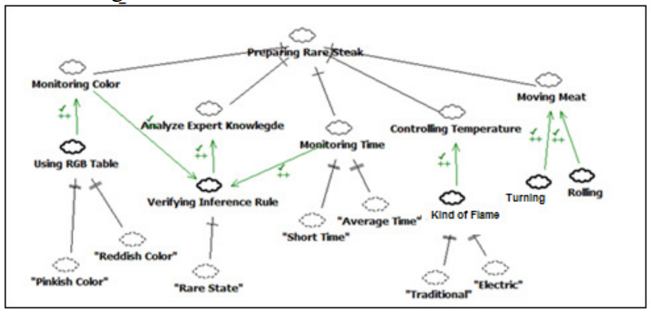


Figure 3. SIG Diagram specifying the adaptive requirement “Prepare Rare Steak”

If the two softgoals above were satisfied, the “Verify Inference Rules” softgoal must be met with the “Rare State” claim softgoal.

B. *Discussion and Analysis of Results*

With the purpose of observing and validating the activities suggested in PERSA process, the theoretical proposal was applied in a case study, which contemplates an adaptive system aiming to determine the degree of understanding, the clarity of activities and the necessary adjustments to improve the activities proposed in PERSA process. PERSA Process was divided in three different stages: analyze the list of requirements, treat adaptive requirements through fuzzy modeling, modeling adaptive requirements through NFR modeling. In the implementation of adaptive requirements specified in the case study, the table Knowledge Base (Table IV) must modify itself at runtime to satisfy the main goal of the adaptive systems, which consists the possibility of alterations at runtime due to the variability in the environmental context.

In the Cook`s Problem, it can be imagined a reading of the tone “Black”: in case it does not fit in any of the inference rules and that would, by approximation lead the adaptive system to an adjustment to this situation by creating a new rule bases on a preexistent one, similar to the color “Black”, thus continuing its running. The new rule would have the following definition: *If Short time and Black Tone, the state of the Steak Burnt*. It is emphasized that the column filled by the specialist do not alter, only the columns with the fuzzy sets. At the end of the case study explanation, there is a satisfactory assessment, because it reached its purpose of specifying requirements for adaptive system. PERSA process specifies adaptive requirements clearly and systematically. Though it is a support technique to software specification demanding the Requirements Engineering to acquire knowledge about Fuzzy Logics and NFR-Framework, it leads to improvements in quality and productivity when developing adaptive systems which justifies the cost of initial investment for the learning of the process. In conclusion, despite performing only few case studies and the need of a wider range of evaluation, based on this initial assessment, the specification outcome is positive, achiever of its goal, confirming that PERSA Process specifies requirements for adaptive systems clearly, effectively and systematically

V. CONCLUSION

This paper presented an approach of requirements specification for adaptive systems, based on the characteristics identified in systemic context with high variability and many fuzzy variables, full of uncertainties as well as the relevant definitions to the adaptive requirements modeling, based on Fuzzy Logic and *NFR-Framework.*

The purpose of this research aimed to assist the existing lack in the requirements specification for adaptive systems. The requirements specification for any type of system is not a trivial task, since it still presents problems identified decades ago. Thus, the adaptive requirements specification, which has special features, such as the possibility of modifying at runtime, makes the challenge even greater.

A. *Main Contributions*

PERSA Process presented in this study aimed to recommend a systematic way to the activities of requirements specification for adaptive systems. The following aspects may be indicated as this study`s main contributions:

- The conception of a requirements specifications process for adaptive systems;
- The creation of a specific requirements documentation for adaptive systems;
- The specification of systemic uncertain and with vague information contexts.

This study limits itself to the requirements specification for adaptive systems by PERSA process. Slightly extending beyond limitation and crossing the border with code/implementation phase, it may be stated that the core of adaptive system is in the creation and management of the Knowledge Base Matrix (as seen in Tables IV and V). The Knowledge Base should be modified at runtime to satisfy the changes in the environmental context, being the main difference of an adaptive system and a conventional one.

B. *Future Works*

This work, through a series of new proposals, can be expanded by further studies. To this end, the following proposals are highlighted:

- Adjustment and inclusion of activities in PERSA process identified by the study of more complex cases;
- Validation of the proposed PERSA process by developing other case studies;
- Development of an automated tool to support and facilitate the use of PERSA process;
- Creation of a repository for storing and retrieving the generated artifacts along the use of PERSA process;
- To perform the next phase of Requirements Engineering (validation), based on the artifact generated by PERSA Process.

REFERENCES

[1] B. H. Cheng, R. Lemos, H. Giese, P. Inverardi, and J. Magee, “Software Engineering for Self-Adaptive Systems: A Research Roadmap”, In: Software Engineering for Self-Adaptive Systems, Springer, 2009, pp.1-28.

[2] A. J. Ramires, B. H. Cheng, and P. K. Mckinley, “Adaptive Monitoring of Software Requirements”, In: Requirements@Run.Time, First International Workshop on, 2010, pp. 41-50.

[3] G. Brown, B. H. Cheng, H. Goldsby, and J. Zhang, “Goal-oriented Specification of Adaptation Requirements Engineering in Adaptive Systems”, In: SEAMS '06: Proceedings of the International ICSE Workshop on Self-adaptation and self-managing systems. ACM, 2006, pp. 23–29.

[4] L. A. Zadeh and R. R. Yage, Fuzzy Sets and Applications: Selected Papers by L.A. ZADEH, Wiley-Interscience, 1987

[5] L. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos, “Non-Functional Requirements in Software Engineering” In: The Kluwer International Series in Software Engineering, Vol. 5, 1999.

[6] L. Baresi, L. Pasquale, and P. Spoletini, “Fuzzy Goals for Requirements-driven Adaptation”, In: 18th IEEE International Requirements Engineering Conference, 2010, pp.125-134.

[7] G. Klir, U. H. St. Clair, and B. Yuan, Fuzzy Set Theory – Foundations and Applications, United States : ed. Prentice Hall, 1987.

[8] W. Pedrycz and F. Gomide, An Introduction to Fuzzy Sets: Analysis and Design, A Bradford Book, 1998.

[9] M. Serrano and J. C. S. P. Leite, “Dealing with softgoals at runtime: A fuzzy logic approach”, In: Requirements@Run.Time, 2nd International Workshop on, 2011, pp. 23 - 31.

[10] S. Liaskos, S. A. Mcllraith, S. Sohradi, and J. Mylopoulos, “Integrating Preferences into Goal Models for Requirements Engineering”, In: 18h IEEE International Requirements Engineering Conference, 2010, pp. 135-144.

[11] P. Sawyer, N. Bencomo, J. Whittle, E. Letier, and A. Finkelstein, “Requirements-Aware Systems: A research agenda for RE for self-adaptive systems”, In: 18th IEEE International Requirements Engineering Conference, 2010, pp. 95-103.

[12] N. A. Qureshi and A Perini, “Requirements Engineering for Adaptive Service Based Applications”, In: 18th IEEE International Requirements Engineering Conference, 2010, pp.108-111.

[13] J. Pimentel and J. Castro, “Specification of Failure-Handling Requirements as Policy Rules on Self-Adaptive Systems”, In: 14th Workshop on Requirements Engineering, 2011. pp. 345 -356.

[14] N. A. Qureshi and A. Perini, “Engineering Adaptive Requirements”, In: SEAMS '09: Proceedings of the ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems. ACM, 2009, pp. 126-131.

[15] N. A. Qureshi, S. Liaskos, and A. Perini, “Reasoning about adaptive requirements for self-adaptive systems at runtime”, In: Requirements@Run.Time, 2nd International Workshop on, 2011, pp. 16 – 22.

Requirements Elicitation Guide for Embedded Systems: An Industry Challenge

Luiz Eduardo Galvão Martins
Institute of Science and Technology
Federal University of São Paulo, UNIFESP
São José dos Campos, Brazil
e-mail: legmartins@unifesp.br

Jaime Cazuhiro Ossada
Technology College of Indaiatuba
Indaiatuba, Brazil
e-mail: Jaime.ossada@fatec.sp.gov.br

Anderson Belgamo
Methodist University of Piracicaba, UNIMEP
Piracicaba, Brazil
e-mail: anbelgamo@unimep.br

Bárbara Stefani Ranieri
Methodist University of Piracicaba, UNIMEP
Piracicaba, Brazil
e-mail: bsranieri@unimep.br

Abstract—**This paper presents GERSE, a guide to requirements elicitation for embedded systems – GERSE is a Portuguese acronym to *Guia de Elicitação de Requisitos para Sistemas Embarcados*. Despite the advances in the area of embedded systems, there is a shortage of requirements elicitation techniques that meet the particularities of this segment. The contribution of GERSE is to improve the capturing process and organization of the embedded systems requirements. The proposed guide was based on a field research with Brazilian developers to find out the state of practice in embedded systems requirements. GERSE had been tested in a case study and had been evaluated by embedded systems engineers. A tool called *ZAKI* was developed to support GERSE and is also presented in this paper.**

Keywords - Embedded Systems; Requirements Elicitation; Requirements Template.

I. Introduction

Currently, the Embedded Systems (ES) projects have been created for a lot of purposes and they are an area with several aspects to be explored. The presence of ES has increased in the last years and they have become almost ubiquitous in segments as industry, commerce and residences [4]. The developed software for ES is becoming more and more complex and sophisticate, of course this increased sophistication has a strong influence in system requirements elicitation and management. In ES context, more than 50% of the problems occur after the system is delivered to the customer [10][15]. However, the described problems are not implementation mistakes, but most of them are requirement issues emerged during the system conception.

The ES are present in our daily life and the trend is to increase in large scale in the next years. Currently, billions of processors have been built a year to supply the ES market [2]. Taking such context under consideration, we present GERSE in this paper as a guide to drive the requirements elicitation for embedded systems. GERSE will support ES developers to create safer, trustworthy, complete, and correct ES using requirements engineering as a basis.

The ES development has grown a lot in the last years, but the industries still have serious problems to define patterns and templates to adequately address the particularities of the requirements definition of ES. The consolidation of the good practices that effectively support the demands of the ES development process is still a great challenge to industry.

This paper is organized as follows: in section two, related works involving requirements engineering to ES are commented; in section three, the phases and activities proposed by GERSE are presented; section four presents a case study and the evaluation of the GERSE, performed by ES engineers; section five presents a software tool to support GERSE; and finally, in section six, some conclusions and future works are pointed out.

II. Requirements Engineering for Embedded Systems

While embedded software is becoming more complex the ES engineers are asking the software engineering community techniques, methods and tools which can help them to improve software quality for ES. On the other hand, software engineering community is recognizing the necessity to adapt the existing methods and to offer new ones to effectively support the particularities of the ES area [15][16][17].

Based on the literature review and the interaction to the ES professionals it is possible to detect just few requirements engineering methods, techniques and tools to address the ES particularities [1][7][9]. For instance, the software development in the automotive industry is a field that brings a great challenge to software engineering, where real time and security requirements come together. For the automotive ES to reach their goals, it is necessary that software control functions work correctly according to strict requirements [5].

Beyond the automotive industry, the ES projects have become larger following the electronic components evolution and consequently making new challenges to requirements engineering. A great challenge is to produce ES with high quality and also supply the market before the system becomes obsolete. As discussed by Cheng and Atlee [8], the development teams must use software engineering techniques and processes to improve the productivity of the developers and the quality of the incoming software. The requirements engineering processes [12][14] help the stakeholders to define what they really need, allowing the suppliers to clearly understand the requirements being implemented in the ES. In this requirements definition process, several professionals with different skills collaborate, such as: users and customers, specific domain experts, marketing specialists, project managers, electrical engineers, mechanical engineers, software engineers and others. For this group, the requirements engineering can offer several benefits, as follows: support to agreements and project planning, shortening the development schedule, offering a consistent basis for deadlines estimation, a baseline for validation and verification, and trustworthy artifacts to drive the ES development.

In Broy's work [5], two phases are suggested to run requirements elicitation:

(i) Pre-phase: the first approximation of the product to be developed; during this phase the strategies and the position of the product in the marketplace are defined. The goals and marketing issues are planned and a document must be written reporting the product constraints and possible alternatives.

(ii) Main phase: based on the results from the pre-phase an agreement among the stakeholders must be done. This agreement is an extensive specification of the technical requirements of the product.

As discussed in [5][15], the conventional methods used to perform requirements engineering are incomplete and do not adequately address the particularities and necessities of ES. The requirements engineering to specify electronics, automotives and other devices that need ES demand adjustments. That is the main point discussed in this work and the motivation to propose GERSE.

To propose the requirements elicitation guide for ES presented in this paper, a literature review about requirements engineering related to many aspects of ES was performed. This review covered the period from 1997 to 2012.. Most of the reviewed works pointed out the issues and difficulties at the early stages of the ES development [3] [6] [13] [19], however, it was not found any suggestion about specific methodologies for capturing and defining requirements, or even a guide to refine and transform the high level requirements - close to customers and users - to technical requirements - close to ES engineers.

III. GERSE: A PROPOSAL FOR A REQUIREMENTS ELICITATION GUIDE

In GERSE elaboration, a field research with 53 professionals that worked with ES in the Brazilian market was initially performed. The goal of this field research about requirements elicitation of the ES was to know the state of practice in Brazil. Therefore, professionals who worked in several segments using ES were invited, most of them being professionals working in industries in São Paulo state. The main segments covered in this research were: automotive systems, industrial automation, home appliance, domotics, medical devices, telecommunication and entertainment.

After the organization and analysis of the field research results, a study about IEEE Std. 830-1998 recommendation [11] and *Volere template* [18] was performed. The *IEEE Std. 930-1998* recommendation suggests how to organize software requirements proposing several generic specification templates. The *Volere* template is a document that suggests a detailed framework to document and organize software requirements. Both *IEEE Std. 830-1998* recommendation and *Volere* template are very known by the requirements engineering community, but they are generic guides for requirements elicitation.

Based on these three elements (*IEEE Std. 830-1998*, *Volere* template and the field research results) groups of activities that compose GERSE were proposed, such set of activities was organized in a way to support the ES engineers to better capture and specify ES requirements.

The main goal of the proposed guide is to help ES engineers during the requirements elicitation process. GERSE leads ES engineers during the elicitation process offering a set of activities that addresses the ES main features. Using GERSE, ES engineers can manage the requirements elicitation process in an organized way. The proposed guide helps the requirements definition allowing its complete specification for products based on embedded technology. GERSE is divided into two phases, named pre-phase and main phase, which are organized in seven categories. These categories are organized in 46 activities, which are responsible to generate the artifacts that will compose the ES requirements. Each activity produces at least one artifact that can be both a document describing a specific feature of the product or a diagram modeling any specific feature. The activities of the pre-phase will help the ES engineers to make the transition from the high level requirements to technical requirements. Figure 1 shows a GERSE overview presenting the categories proposed to each phase.

GERSE is divided into two phases: pre-phase e main phase. During the pre-phase the activities were gathered into three categories: Context Organization, Stakeholders Definition and High Level Requirements Elicitation. In the main phase, the activities were gathered into four categories: Definition of Hardware Requirements, Definition of Software Requirements, Identification of Quality Metrics and Identification of Production Requirements. Considering the activities of all categories GERSE offers 46 activities to perform a complete requirements elicitation of the ES. Each category has specific goals supported by activities that should help the ES engineers to produce useful artifacts to compose the requirements specification of the ES to be developed.

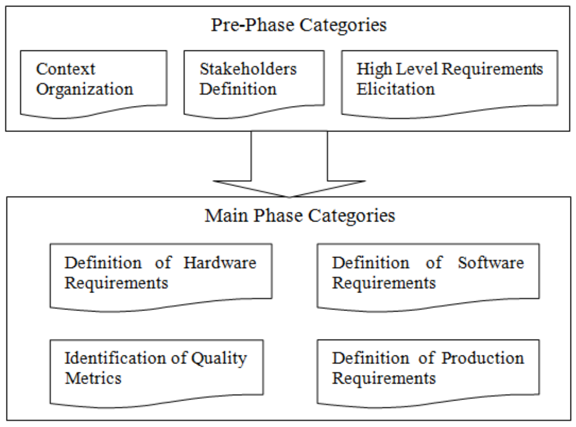

Pre-Phase Categories
Context Organization
Stakeholders Definition
High Level Requirements Elicitation
Main Phase Categories
Definition of Hardware Requirements
Definition of Software Requirements
Identification of Quality Metrics
Definition of Production Requirements

Figure 1. Phases and categories supported by GERSE.

During the pre-phase, the requirements that will help the ES engineers to understand the system to be developed are captured, such requirements define the system basic features, purposes, and goals. The final artifact obtained using GERSE set of activities is a high level requirements specification, which defines all the ES characteristics, referring to mechanical, electrical and functional aspects plus the system cost overview, prototype model and all functional and non-functional requirements. It is important to observe that ES non-functional requirements are different from those usually managed in conventional systems. For example, energy consumption is an ES specific non-functional requirements.

When the pre-phase requirements are gathered, it is necessary to transform them into technical requirements, such transformation will enrich the requirements with more details. In this process, the category "High Level Requirements Elicitation" has a very important role because the requirements obtained from this category will be the requirements core to be transformed into technical requirements. After all activities suggested in GERSE are performed, the ES engineers gather a large set of functional and non-functional requirements that specify the main features of the ES. The cost to gather such requirements documentation is low when GERSE activities are followed by the ES engineers. This documentation facilitates the project development in parallel ways: one team developing the hardware and other team developing the software, turning the ES development faster to answer the time to market.

IV. CASE STUDY AND GERSE EVALUATION

In this section, a case study is presented using GERSE. The guide was instantiated to produce the requirements elicitation of a digital chess clock used in professional chess tournaments. The purpose of this experience was to evaluate GERSE in a real situation choosing a specific product and eliciting ES requirements that must control such product. The case study shows some artifacts generated using GERSE during the requirements elicitation process.

Table 1 presents the results gathered from the activities performed in the category "Stakeholders Definition". The identified stakeholders include: commercial department, marketing department, components suppliers, chess referees, other manufacturers, chess players, and hobbyists. The evolvement degree and influence degree on the project were defined for each type of stakeholders.

TABLE I. RESULTS GATHERED IN THE ACTIVITIES FROM THE CATEGORY "STAKEHOLDERS DEFINITION"

Activity	Output Artifacts
Definition of key stakeholder	**Commercial Department:** Involvement level: low Influence on project: Approval of final costs. **Department of Marketing** Involvement level: High Influence on project: approval of the characteristics of packaging, reliability, performance, usability and action buttons. **Component manufacturers and suppliers** Involvement level: High Influence on project: collaborate by providing technical specifications of the components to engineers to make better use of components to be used in product development.
Determine domain experts stakeholders	**Chess referee** Involvement level: High Influence on Project: help understand the official rules of chess game according to the FIDE (World Chess Federation).
Identify stakeholders against the project	**Competitors' manufacturer** Involvement level: High, because the clocks available on the market by manufacturers are used as references for comparison of new product.
Characterize User Profiles	**Professional chess players** Profile: Users accustomed with digital chess clock available in the market. Use the clock to study and compete. **Hobbyists** Profile: users more accustomed to analog clocks. Use the clock to study, and eventually in competitions.

Figure 2 presents a suggestion for the design of the case, as well as the keys and the chess clock display, this prototype is resulted from the activities in the category "High Level Requirements Elicitation". This prototype was based on market analysis and a requirements elicitation process performed with professional chess players, which pointed out the main functions that a digital chess clock has to offer for the users, especially for those who use it in professional tournaments.

The complete requirements specification of the digital chess clock and GERSE documentation were sent to four ES engineers to evaluate the proposed guide, the evaluation was performed based on a survey. The ES engineers' expertise was in automotive systems, medical devices and entertainment areas. The survey was composed by twenty one questions based on Likert's scale [20]. The case study results, GERSE documentation and the survey were e-mailed to the ES engineers. The answers to the survey allowed a realistic evaluation of GERSE viability.

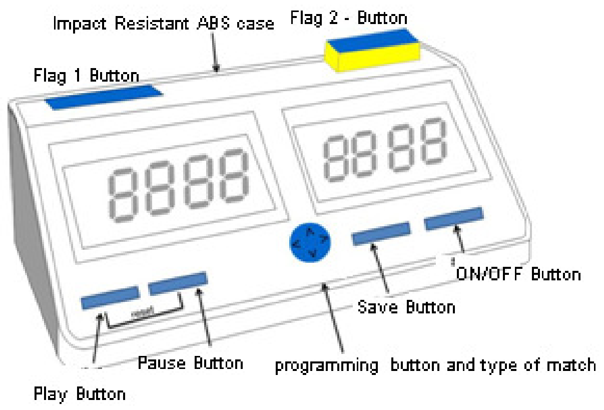


Figure 2. Initial product prototype showing the main functions - gathered in the activities performed from the category "High Level Requirements Elicitation".

Table II presents the results gathered in the activities performed from the category "Definitions of Hardware Requirements". In this table, the main hardware requirements of the digital chess clock are presented, which specify the sensors, interaction displays, keys, external communication interface, hardware interruptions and microcontroller necessary to build the product. Considering the unique aspects of this project it was not necessary to specify actuators.

TABLE II. RESULTS GATHERED IN THE ACTIVITIES FROM THE CATEGORY "DEFINITION OF HARDWARE REQUIREMENTS"

Activity	Output Artifacts
Determine sensor	**01 Humidity sensor** Function: continually check the internal moisture of the product to avoid damage to the components. Type: analog. **01 Temperature sensor** Function: check the value of the internal chassis temperature, to ensure that it will not exceed the working temperature range of the internal components. Type: analog.
Delimit the actuators	Not applied to this product.
Clarify user interaction	**02 LCD graphical with 128 x 64** Function: display the playing remaining time for each player (update the display in real time). **01 Buzzer** Function: warning to the end of settings, end of the game and battery low level.
Characterize hardware interruptions	**Temperature Range** Function: sound and light warning should be issued if the temperature is outside the ranges of acceptable values. **Humidity range** Function: sound and light warning should be issued if the humidity is outside the ranges of acceptable values. **Battery** Function: sound and light warning should be issued if the battery low level. **Pause** Function: by pressing the"pause" the time count both counters should be frozen.
Identify the action buttons	**02 on/off switch with lock** Function: activation of the stop watch time player who makes the move (and freezing the timer player that does the move). **01 Button type joystick** Function: button for programming should have 4 positions (left, right, up, down) and a central (enter), for control and navigation mode for programming. **04 Buttons without locking** Function: buttons to pause, start, save programming, turn off and on the clock.
Specify the memories	**01 PROM memory** Function: storage modalities of game time. **01 Flash memory** Function: store setup(variable) of types of playing time.
Define external communication ports	**USB Port** Function: connection to external board for automatic storage of moves.
Fix component requirements	**AC / DC Adapter** Function: battery charging
Specify the requirements for layout of controller board	The layout of the printed circuit board to be double sided to contribute to miniaturization of the enclosure.
Defining the parameters of legacy hardware	Not applied to this product.
Demarcating the parameters of special COTS	Not applied to this product.
Identify microcontrollers	PIC PIC18F4550-I/P 32 KB/2048 RAM 35 I/O microcontrollers with USB support.

All ES engineers evaluated GERSE as a useful guide for ES requirements elicitation stating that such guide is easy to use and contributes to increase the ES development quality. Table III shows GERSE general evaluation results. It is possible to observe that GERSE was well evaluated, especially the aspects concerned to clearness, easiness of use and the contribution to improve the quality of requirements elicitation. The ES engineers considered the guide easy to use and it supports their requirements elicitation necessities. The completeness is an issue that must be improved in GERSE.

TABLE III. GERSE GENERAL EVALUATION

Questions	Totally agree	Partially agree	Partially disagree	Totally disagree
The presented guide is clear enough to be used in an embedded systems design for small and medium businesses.	50%	50%	0%	0%
The presented guide is complete and meets the needs of embedded systems projects for small and medium businesses.	50%	25%	25%	0%
Would adopt the presented guide for requirements elicitation on future projects.	50%	25%	25%	0%
The presented guide is easy to use.	50%	50%	0%	0%
The presented guide contributes in improving the quality of embedded systems development.	50%	50%	0%	0%
The presented guide meets your needs requirements definitions in embedded systems projects.	50%	50%	0%	0%

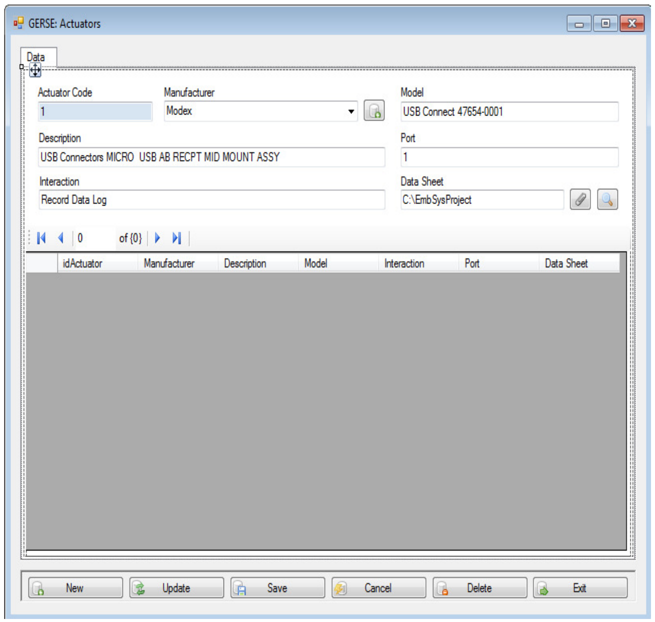
V. *ZAKI*: A COMPUTATIONAL SUPPORT TO GERSE

The adoption of any software process can be facilitated by the use of computer support. In this sense, a tool called Zaki [21] was developed, to support GERSE activities and the requirements elicitation process for embedded systems.

Zaki tool is divided into two modules, according to GERSE phases (pre-phase and main phase), supporting activities like requirements elicitation, analysis and management for embedded systems. Zaki tool was developed using .NET platform (C# language) and the SQL Server database.

During the pre-phase, Zaki tool supports functionalities related to manage information about project guidelines and main product features, development organizational impact and target audience. During the main phase, Zaki tool is divided into three modules: Definition of Hardware Requirements, Definition of Software Requirements, and Identification of Quality Metrics.

Specifically related to Definition of Hardware Requirements, Zaki tool converts high level requirements to technical ones, allowing the definition of sensors, actuators, memory, microcontrollers, legacy hardware and other requirements associated to hardware components. Besides, it is possible to choose COTS (Commercial Off-The-Shelf) to be used in the embedded systems. Figure 3 presents a user interface of Zaki tool responsible to record actuators related to the embedded system project.



Actuator Code: 1
Manufacturer: Modex
Model: USB Connect 47654-0001
Description: USB Connectors MICRO USB AB RECPT MID MOUNT ASSY
Port: 1
Interaction: Record Data Log
Data Sheet: C:\EmbSysProject

0 of {0}

idActuator	Manufacturer	Description	Model	Interaction	Port	Data Sheet

Figure 3. User interface of *Zaki* tool to manage actuators.

Aiming to perform a feasibility study of Zaki tool, three requirements engineers - with over two years of experience in the area - were asked to perform the requirements elicitation and specification for an embedded system to a data logger device. The main goal of such device is monitoring and collecting environmental data, including temperature, atmospheric pressure, humidity, rainfall, wind speed, and others.

TABLE IV. ADDRESSING QUESTIONS TO THE ELICITATION PROCESS SUPPORTED BY ZAKI

Questions	Totally agree	Partially agree	Disagr
The tool meets the goals of the requirements elicitation process for embedded systems.	100%	0%	0%
The tool facilitates the process of requirements elicitation, assuring quality of project and time reduction.	100%	0%	0%
The tool organizes information about the project and ensures an efficient requirements elicitation process.	25%	75%	0%
The tool supports a complete requirements elicitation process, ensuring the completeness of the project goals.	100%	0%	0%

The evaluation was performed by the filling of a questionnaire with 15 questions - 13 objective questions and 2 personal observations. The goal of evaluation was to analyze the use of Zaki tool to identify improvements and non compliances.

According to Table IV, the requirements engineers were unanimous in stating that the tool supports the requirements elicitation process, facilitating and reflecting in time savings and quality of the embedded systems project. However, interfaces improvements must be performed to facilitate the usage of Zaki tool.

VI. Conclusion and Future Work

GERSE proposed activities support engineers guiding ES requirements elicitation and specification, which help them to produce an organized, easy to understand and complete requirements document. According to the evaluation performed by ES engineers, GERSE reaches the goal of being itself a consistent guide for ES requirements elicitation. One of this work's relevant contribution is to narrow the gap between Software Engineering – specifically concerned to Requirements Engineering - and ES engineering. The requirements elicitation for any kind of system is not a trivial job. Particularly for ES, there are a lot of specific issues to be managed, for instance: real-time requirements, energy consume control, hardware constraints – sensors, actuators, memory and microcontrollers – short window of *time-to-market*.

GERSE evaluation performed by ES engineers with expertise in several areas of application, as automotive, medical devices and entertainment, pointed out that the guide is clear and easy to understand. But the evaluation also reveals that some aspects can be improved such as GERSE completeness, specially the activities concerned to quality metrics and production requirements. These issues are going to be treated in future works. In general, GERSE was considered satisfactory contributing to fulfill the existent gap in the early stages of an ES project. GERSE contributes to decrease the occurrence of faults, errors and mistakes that are very common during the ES requirements capturing. A mature requirements elicitation process can be reached using GERSE, which supports the transition of high level requirements to technical ones. This paper also presented *Zaki,* a tool to support GERSE adoption, since it can assist ES engineers to better manage the requirements gathered when GERSE is running.

References

[1] A. Post, I. Menzel, J. Hoenicke, and A. Podelski, “Automotive Behavioral Requirements Expressed in a Specification Pattern System: a Case Study at BOSCH”, Requirements Engineering, Springer-Verlag, vol. 17, 2012, pp. 19-33.

[2] M. Aoyama, “Persona and Scenario Based Requirements Engineering for Software Embedded in Digital Consumer Products”. 13th IEEE International Conference on Requirements Engineering, 2005, pp. 85-94.

[3] A. Aurum and C. Wohlin, “Requirements Engineering: Setting the context”. In: Aurum C. & Wohlin C. (Eds), Engineering and managing software requirement. Springer. Berlin, Germany, 2005, pp. 1-15.

[4] J. Boulanger and D. van Quang, “Experiences from a model-based methodology for embedded electronic software in automobile”, (ICTTA) 3rd International Conference on Information and Communication Technologies: From Theory to Applications, 2008, pp. 1-6.

[5] M. Broy, “Requirements Engineering for Embedded Systems”, Workshop on Formal Design of Safety Critical Embedded Systems (FemSys), Munich, Germany, 1998.

[6] R. Cancian, M. Stemmer, and A. Frohlich, “New Developments in EPOS Tools for Configuring and Generating Embedded Systems”, Proceedings of the 12th IEEE International Conference on Emerging Technologies and Factory Automation, Patras, 2007, pp. 776-779.

[7] H. Chae, “The Partitioning Methodology in Hardware/Software Co-Design Using Extreme Programming: Evaluation through the Lego Robot Project”, Proceedings of the sixth IEEE International Conference on Computer and information Technology, 2006, p. 187.

[8] B. Cheng and J. Atlee, “Research Directions in Requirements Engenneering, Future of software Engeneering”. IEEE Future of Software Engineering (FOSE’07), 2007, pp. 285 – 303.

[9] E. Sikora, B. Tenbergen, and K. Pohl, “Requirements Engineering for Embedded Systems: an Investigation of Industry Needs”, Proceedings of the 17th International Working Conference on Requirements Engineering: foundation for software quality (REFSQ'11). Springer-Verlag, Berlin, Heidelberg, 2011, pp. 151-165.

[10] B. Graaf, M. Lormans, and H. Toetenel, “Embedded Software Engineering: the State of the Practice”, IEEE Software archive, vol. 20, no. 6, 2003, pp. 61-69.

[11] IEEE Computer Society Software Engineering Standards Committee, “IEEE Recommended Practice for Software Requirements Specifications”. IEEE Std 830-1998.

[12] H. Hoffmann and F. Lehner, “Requirements Engineering as a Success Factor in Software Projects”. IEEE Software, vol. 18, no.4, 2001, pp. 58-66.

[13] L. Jiang and A. Eberlein, “Selecting Requirements Engineering Techniques Based on Project Attributes - A Case Study”. Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer Based Systems, 2007, pp. 269 – 178.

[14] G. Kotonya and I. Sommerville, Requirements Engineering: Processes and Techniques. John Wiley and Sons, 1998.

[15] P. Liggesmeyer and M. Trapp, “Trends in Embedded Software Engeneering”, IEEE Software Magazine, v. 26, n. 3, 2009, pp. 19-25.

[16] E. Nasr, J. Mcdermid J, and G. Bernat, “Eliciting and Specifying Requirements with Use Cases for Embedded Systems”, Proceedings of the 7th International Workshop on Object-Oriented Real-Time dependable systems (WORDS), 2002, pp. 350 – 357.

[17] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, “Software Engineering for Automotive Systems: A Roadmap Future of Software Engineering”, In: International Conference on Software Engineering - Future of Software Engineering (FOSE), 2007, pp. 55 -71.

[18] S. Robertson and J. Robertson, Mastering the Requirements Process, Addison-Wesley, 2nd edition, London, 2006.

[19] F. Vahid and T. Givargis, Embedded System Design: A Unified Hardware/Software Design. John Willey & Sons, 2002.

[20] R. Likert, “A technique for the measurement of attitudes”. Archives of Psychology, 1932.

[21] J. C. Ossada, L. E. G. Martins, A. Belgamo, and B. S. Ranieri, “GERSE: Guia de Elicitação de Requisitos para Sistemas Embarcados”, XII Workshop on Requirements Engineering (WER), Argentina, 2012, pp. 1-14.

Separation of Concerns and Code Enhancement: Aspect-oriented Programming Versus Customization Approach Followed in Open Source Software

Sidra Sultana
Department of Computer Software Engineering
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
sidra.sultana88@gmail.com

Fahim Arif
Department of Computer Software Engineering
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
fahim@mcs.edu.pk

Abstract— **In order to facilitate the separation of concerns and code enhancement without modifying the original code, open source software (OSS) offers a package containing the core code. Depending upon the design or architecture pattern followed in the specified package, the ways to facilitate code enhancement are provided. Hook Architecture is followed in Wordpress, Drupal, etc., in customizing plugins or modules, and Model View Controller (MVC) pattern is followed in Joomla, open source content management systems. Aspect-oriented Programming (AOP) is a programming paradigm that addresses the same code scattering and code tangling issue, and thus, ensure code enhancement without modifying the core code. The research question is whether AOP supports the separation of concerns and allows the enhancement in functionality without modifying the core code; then, hook architecture and other open source customization patterns are there to facilitate the goal. What different features does it offer, as compared to AOP? This research paper differentiates between the separation of concerns and code enhancement addressed by OSS and AOP.**

Keywords-Aspect-oriented Programming (AOP); Open Source Software (OSS); Advice; Joinpoint; Pointcut; Hook Architecture; MVC pattern; Aspect-oriented Model View Controller (AOMVC)

I. INTRODUCTION

Aspect-oriented Programming (AOP) [1] is a programming paradigm that complements Object-oriented Programming (OOP) [2] by separating *concerns* of a software application to improve modularization. The separation of concerns (SoC) aims at making software easier to maintain by grouping features and behavior into manageable parts, which all have a specific purpose and business to take care of. It is the decomposition approach followed in the conventional modular programming that leads to code tangling (code mingling) and code scattering (replication and duplication of same code chunk at many places).

Third party tools, off-the-shelf components, and open source modules are there to be used by the current application; if the application is flexible enough to utilize it without modifying the core code and by simple joining the new functionality from a point where changing (adding/removing) additional code is easier to maintain.

Thus, an effortless and unified approach is offered by AOP in terms of making the dynamic switching of complete features along with providing the conciseness, evolution, and testability. Aspect-oriented approach focuses on the argument related to the maintainability and readability of the constructed software.

Section II offers a brief literature survey. The comparative analysis is performed in Section III. Research Results are presented in Section IV. Section V provides a discussion. Conclusion is given in Section VI.

II. LITERATURE SURVEY

AOP is designed to formulate code easier to query about, trace, develop, enhance, maintain, and modify certain verity of application code. For the sake of validating these potentials claimed by AOP and to verify the impact of AOP on the program structure, Robert et al. conducted two investigatory experiments [1]. AspectJ version 0.1 [14] was the language in which the requirements are implemented to trace change and debugging process supported by AOP. Developer’s ability to trace and then resolve the issues (programming fault) of the multi-threaded program is analyzed in the very first experiment. In the other experiment, existing distributed system is focused on checking the ease in change management provided by the AOP.

A. *Modularization in AOP*

Kiczales et al. [2] have familiarized AOP for providing more organized and well managed way of capturing the code while enhancing the scope of the program concerns. Software programmers explicitly manage the separation of some concerns within the code by the help of built-in functionalities provided by the selected programming language. Explicit language support is provided by AOP to help functional decomposition in the program and to be well modularized upon the design decisions.

B. *Usability of AOP*

Usability and usefulness of AOP are well proved in the experimental results [3]. The core code that is functionally decomposed and aspects’ interface has some characteristics highlighted by the experiment, to show that programming benefits can be accrued best with the understanding of it. Vital feature as per the completeness point of view of AOP approach is that, it is beneficial in totality [4]. This refers to the fact that partial benefits cannot be extracted by the partial implementation of separation of concerns. Well defined scope of the aspect effected across the boundaries, is necessary to provide the refined (narrowed) scope of the aspect without digging deep the core code for extensive analysis. Thus, when the separation is more complete, i.e., interface is narrow, only then the AOP approach will be more promising [5] [6].

C. *Design Quality in AOP*

With regards to the design quality and software development efficiency [7], a web based system is developed to empirically study its behavior in both AOP and OOP fashion. The study reveals that if the number of subjects undertaken in the experiment increases, then benefits offered by AOP will be much more as compared to those underline in the present study. To produce high quality, design aspects are very vital so, Madeyski et al. [8] aimed at providing empirical evidence of the impact of AOP on design quality metrics and software development efficiency.

III. Comparative Anaysis

In order to facilitate the separation of concern and code enhancement without modifying the original code, OSS provides with a package containing the core code. Depending upon the design or architecture pattern followed in the specified package, the ways to facilitate code enhancement are provided. Hook Architecture is followed in Wordpress, Drupal, etc., in customizing plugins or modules, while MVC pattern is followed in Joomla, FLOW3, etc., open source content management systems. AOP is a programming paradigm that addresses the same code scattering and code tangling issue and thus, ensure code enhancement without modifying the core code. The research question is whether AOP supports the separation of concerns and allows the enhancement in functionality without modifying the core code; then, hook architecture and other open source customization patterns are there to facilitate the goal. What different features does it offer, as compared to AOP?

For the comprehensive analysis, three aspects are implemented in FLOW3 (an open source framework) to address all cross cutting concerns in components of MVC.

For potential cross-cutting concern in Model Class, Logging Aspect is used to log the delete details, in other case; it can be mistakenly added as a part of business logic in Model class of the package.

To address potential cross-cutting concern in View Class, Flash Message Aspect is used to inject html element (i.e., styled div) with specific list of actions, thus addressing the cross cutting concerns at interface level or View class of the package.

For potential cross-cutting concern in Controller Class, Manipulation Aspect is used to provide control access for number of controller's actions so in terms of addressing control flow, manipulation aspect resolves cross cutting concerns in Controller Class.

Kato et al. [21] also presented the Context-Oriented Programming implementation along with the OOP and AOP comparison but lacking the comprehensive metrics analysis. The novelty of the conducted research lies in the wide domain discussion of the concerned problem in functional and non-functional requirements domain like maintainability, re-usability, scalability, code organization, dynamics, etc.

This section differentiates between the separation of concerns and code enhancement addressed by OSS and AOP and thus, giving an insight of AOMVC and MVC cross-cutting concerns resolved by MVC.

A. *OSS*

OSS like CMS [8] or frameworks provide with the general package containing backend (administrator view) and front end (user view) of the application. Some of the cross cutting concerns like security (Manipulation Aspect), logging (Modeling Aspect), flash messaging (View Aspect) etc., are addressed by the CMS and frameworks like Joomla, Drupal, Wordpress, YII, Zend, Virteom, Magento, Oscommerce, etc.

Almost all OSS followed certain programming approaches for handling the separation of concerns and demotivates modifying the core code. Mostly MVC or Hook Architecture is followed to code custom components, modules, or plug-ins. It helps in enhancing the application functionality in a flexible adding/removing way.

B. *AOP*

"Separation of concerns" principle has been used for many years by software engineers to handle the software system's development [9]. Software programmers explicitly manage the separation of some concerns within the code by the help of built in functionalities provided by the selected programming language. Explicit language support is provided by AOP to help functional decomposition program and to be well modularized upon the design decisions.

AOP is made for code enhancement, so that the cross cutting code related to the design decision is not dispersed throughout the program rather it is expressed in a separate set of coherent code chunks [10]. AOP owns a better way of modularizing cross-cutting concerns, resulting in the more readable and less complex developed system implementation.

C. *Cross Cutting Concerns*

Allowing the modularization of the concerns that usually cross-cut in the object-oriented way of programming application [11], AOP resolved number of programming issues encountered by OOP like code tangling and code scattering, all as result of cross-cutting concerns.

Aspects are declared by using around, after and before advices for the retrieval of properties and intercepting settings.

D. *Code Enhancement in OSS*

In order to facilitate the code enhancement without modifying the original code, OSS provides with a package containing the core code. Depending upon the design or architecture pattern followed in the provided package, the ways to facilitate code enhancement are specified [12]. Hook Architecture is followed in Wordpress, Drupal, etc., customizing plugins or modules, while MVC pattern is followed in Joomla, etc., Open Source CMS.

E. *Code Enhancement in AOP*

Code scattering and code tangling are not the only results of implementing security concerns in an application - by following OO approach - but it also because the weaker existence of the security related issues. AOP addresses this code scattering and code tangling issue hence, advocating an improvement in dealing these issues previously in OO way. A number of reasons are there for showing weaker enforcement of security including programming error, inherit design of the system etc.

Conventional software engineering practices failed to modularize cross-cutting concerns and Aspect-oriented Software development offsets this limitation of current software engineering constructs. The advice injected in the point-cut expression is to be bonded after, before or around the code. Also, wildcards (.*) can be used to bind advice with number of join-points. This flexibility of hooking the code at number of places creates the difference and provides an edge to the AOP paradigm.

F. *AOMVC*

MVC refers to modularizing the application in terms of separating the layers of Control flow and management (i.e., Controller), Interface Design (View) and Database interaction (Model) [13]. MVC framework, in the domain of J2EE [14], has cross cutting concerns throughout the multiple modules (e.g., validation transaction, logging, etc.).

MVC framework is the well-known layered architecture but it has greater limitations and architectural constraints in dealing with cross-cutting concerns. These overlapping concerns lead to code confusion, code tangling and code scattering and finally, result in the difficulty of system maintenance and extensibility. AOP addresses all these problems in every layer of abstraction, i.e., Model, View and Controller. Aspects can be defined to modularize such concerns. All such concerns are well defined by the aspects of AOP.

G. *MVC cross-cutting concerns and AOP*

The three potential cross-cutting concerns that address almost all components of MVC are presented.

a. *Potential cross-cutting concern in Model Class*

Logging Aspect is used to log the delete details and hence can be mistakenly added a part of business logic in Model class of the package.

b. *Potential cross-cutting concern in View Class*

Flash Message Aspect is used to inject html element (i.e., styled div) with specific list of actions, thus addressing the cross cutting concerns at interface level or View class of the package.

c. *Potential cross-cutting concern in Controller Class*

Manipulation (security) Aspect is used to provide control access for number of controller's actions so, in terms of addressing control flow, manipulation aspect resolves cross cutting concerns in Controller Class.

Thus, by extracting the different cross-cutting concerns from the model, view and controller component of the MVC model, an aspect layer is to be composed to weave with the core functionality.

IV. RESEARCH RESULTS

Some of the factors that distinguished the contribution of AOP and OSS for separation of concerns and code enhancements are: point of access, code management, development time, line of codes, and functional breakdown, etc. These qualitative and quantitative factors that contribute in the estimation of software metrics are analyzed in this section.

A. *Point of Access*

In case of AOP, aspect classes with variety of advices are defined to be injected at different levels of code. For example, this injection of the wildcard \before ("method (.*Controller->.* Action ())") to all controllers actions will bind the particular advice with all actions of every controller. \before ("method (studentController->.*Action ())") this one-to-many injection will affect all actions of student controller only and \before ("method (studentController->registerAction ())") this one-to-one injection will bind the advice to registerAction of the studentController and for all three injection types, advice will be bound before the action's code. This single class is the single point of access for all related code management in terms of adding and removing the aspect's advices.

For OSS, customization is to be ensured by coding plugins, components and modules as per the coding conventions of the selected OSS. In that case modifications are to be managed in multiple files and thus, there are multiple points of code access that increases the complexity measure.

B. *Separation of Cross-cutting Concerns*

AOP is designed for handling cross-cutting concerns and thus, resolving them by addressing the code tangling and code scattering issues. Code Tangling refers to the phenomena where the concerns are interwoven with each other in a module. Code Scattering occurs when the concerns are dispersed over many modules. It results in a typical design problem of high-coupling and low cohesion. All the components that are specifically fragmented using the traditional techniques for highlighting their role as a cross-cutting concern, should be well evaluated. For instance, if a logging functionality is implemented in an aspect-oriented way then in large number of modules invocation to the logger necessitates being present in the model.

The interesting insight of the aspect-oriented implementation is that along with providing the modularized solution to cross-cutting concerns there is no negative effect on software size and system modularity with AOP implementation. If any particular task is to be performed at a lot of places, then that particular functionality, for instance logging, will be the part of the application domain logic. All of the functional dependencies related to logging would be then injected into the model. Logging is not the domain model logic, neither its view nor controller. So, it does not fit in any layer of MVC. Aspect logging is the non-functional requirement and an example of cross cutting concern. Therefore, such concerns should be implemented in a separate layer, i.e., the Aspect Layer. Hook Architecture is followed along with MVC to run the code side by side in most of the AOP applications.

Separation of cross-cutting concerns is not addressed in OOP, thus OOP with AOP is suggested for better modularization and code optimization.

C. Change Management

Due to singularity of Aspect Class, maintainability and change management is easy for AOP. For OSS, plugins and components have multiple files, so need to track all related code in case of any required modification.

Insertion and deletion in case of OSS is also complex like change management and thus affected other related metrics like development time, line of codes, coupling and cohesion etc.

D. Code Enhancement

In case of OSS convention modular code enhancement, scope of the customized or enhanced code is specific to that particular module for customization of the package. And the defined code has a limited impact on the package. For hook architecture (Wordpress and Drupal, etc.), flexibility of hooking enhanced code is ensured through a single function definition instead of multi-files modules or components. But the impact of the hooked functionality is at a single code point and there is no way to hook the same code to multiple points of the package's core code.

Wildcard (.*) access in case of AOP advice binding enhanced the impact to advice to wide variety of code clones. For example, this injection of the wildcard \before ("method (.*Controller->.*Action ())") to all controllers actions will bind the particular advice with all actions of every controller.

E. Development Time

Aspects developer requires one time focus to learn the aspects implementation and once learned she can bind advices of aspects to any desired code clone. As no knowledge of the current system is required for aspects implementation, the development time is optimized by aspects customization and the development time is focused on required functionality instead of replicating and testing the same code at number of points.

For OSS customization, knowledge of the current system is required, so development time is also spent on related modules. As per the OSS architecture and conventions, there are variable maintenance time issues.

F. Line of Codes

In order to measure the size of the set of instructions – the computer program – there is a metric named line of code LOC, which simply shows the count of the number of code lines of program. Maintainability, programming productivity and effort to be required for developing a program are predicted by LOC. As the cross cutting code is resolved at a single point, line of codes are limited. The same code needs to be coded at all required points, so, line of codes are more as compare to that in AOP.

For instance, there is a requirement of making a detailed entry with timestamp in a logger file whenever any record is deleted. For this simple requirement, wherever delete code is written in the package OSS customization approach will handle the case by coding a plugin, component or module to log the details separately for every code. Thus, if the modified functionality is 'm' and number of clones to be modified is 'n', then the m*n is the number of code lines (LOC) increased in case of OSS customization approach.

In case of AOP, LOC increases by 'm*1', meaning that 'm' lines are added in the original LOC. If there is a single point of change, then, the OSS and AOP approaches are equally to adopt but in common practices logging related codes are required at number of joinpoints. This refers to the strong adoptability of the AOP for large scale projects. In the light of this calculation, it revealed that the usability of aspect-oriented technique directly depends upon the size of application. In case there is a large number of code clones then, the AOP will help in reaping maximum time saving benefits whereas the development speed decreases when this technique is used for small number of code clones.

G. Direction of Functional Breakdown

For a student manager, customization in terms of adding student registration functionality, the direction of functional breakdown varies as per nature of the functionality to be focused. For instance, student registration comprises of two main modules, i.e., Managing Student Bio data and Managing Student Courses. Courses Manager is further divided into content manager and batch manager with course information. All these managers are the functional breakdown of registration manager in top to down direction and thus, will be implemented by OSS way of customization as modularization is done in a vertical fashion.

In case of displaying a flash message on every successful insertion of record in registration manager, advices need to be defined to manage the case in AOP way. For AOP, cross-cutting concerns are handled in the horizontal fashion, i.e., left to right.

Thus, a combination of AOP and OSS customization will be used where the cross-cutting concerns are implemented in AOP to manage code maintainability in single file and other particular module functionalities are implemented in OSS modules, plugins or components.

Summary of these qualitative and quantitative factors that contribute in the estimation of software metrics are tabularized in Table 1.

TABLE I. AOP VS OSS

	AOP	OSS
Point of Access	Single File	Multiple Files
Separation of Cross-cutting Concerns	Resolved	Not Addressed
Code Enhancement	Wide Impact	Limited
Change Management	Easy	Complex
Development Time	Optimized	Increased
Line of Codes	Optimized	Increased (Replication in case of cross-cutting concern)
Direction of Functional Breakdown	Vertical	Horizontal

System having cross cutting concerns can be successfully handled through AOMVC using AOP techniques. AOMVC creates an additional layer of aspects and then declared the aspects in the configuration file in order to provide scalability, maintainability and refined modularization within the system. Also, wildcards can be used to bind advice with number of join-points. This flexibility of hooking the code at number of places creates the difference and provides an edge to the AOP paradigm.

V. DISCUSSION

The potential benefits as per the system's features offered by the AOP approach include the simplicity, readability and modularity. This way, the created system with improved software development efficiency works faster than its object-oriented version.

A. *Code Reuse*

Reusability of the code refers to the phenomena of writing the code once and using it later on number of occasions as per the scenario defined. Once a code is defined and as per its invocation, it gets weaved and called on multiple locations. Hence, the code duplication is reduced manifold. In case of Manipulation aspect the reusability measure is too high to affect number of code clones. Thus, through single point of access, code gets reused and maintained.

B. *Maintainability*

System gradation is a part of every real world application. Code once developed has to be maintained and to ensure configuration management application maintainability is a vital concern for meeting user's needs. Instead of tracing the code in each and every file for the modification or deletion purpose, AOP offers a woven point defined as per language selection in XML, PHP, JAVA, .NET etc., in the declarative way, in order to delete the cross-cutting concern if it is no longer in need, which progresses the maintainability of the system compared with traditional methods - one by one steps to locate the code.

C. *Scalability*

Through scalability, demand for the change in functionality of the original system is facilitated. New functional requirement proposed by the user is coded as an aspect in the form of new feature, specified in the configuration files, woven or bind in a respective point instead of updating number of files required to be modified. Hence, aspects provide scalability for a large amount of changes in the current system in the way to incorporating user's emerging requirements with the passage of time.

D. *Reduced Development Time*

As the line of code is decreased in case of using OOP with AOP, so the development time gets reduced. In case there is a large number of code clones (as in case of Manipulation Aspect) then the AOP will help in reaping maximum time saving benefits whereas the development speed decreases when this technique is used for small number of code clones.

E. *Code Organization*

Cross-cutting concerns of logging, flash message and manipulation are kept aside from Model, View and Controller classes in case of coding aspects for logging, flash message and manipulation functionality. Thus, the domain logic is not confused with the supporting domain logic (logging entry in file or database) in case of logging aspect implementation.

F. *Changeability*

Request for change in web application is too common. With the advent of technology changeability should be offered by the web development. Code once developed has to be maintained and to ensure configuration management application maintainability is a vital concern for meeting user's needs. If in case of Logging Aspect, instead of recording entry in file, requirement got changed to record entry in database then a single line of aspect get replaced instead of replacing code in every related file in case of OOP without AOP.

G. *Extensibility*

Aspects provide scalability for a large amount of changes in the current system in the way to integrating user's evolving requirements with the project advancement. In case of Logging Aspect, if along with recording deletion time in file, recoding an entry in database is required then a single line code at one place need to be added in the logging aspect class.

H. *Dynamics*

Dynamics refers to the enabling and disabling of the aspects. If the injected functionality is no more required then the aspect injection code can be commented. In case of Logging Aspect, if the logging of the delete record is no more required then single line code of recording time of the delete can be commented. Similarly, if the Flash Messages are not to be injected then the code can be commented and same is the case for manipulation aspect.

VI. CONCLUSION

OSS customization mostly follows OOP. Replacing the OOP by AOP was an obsolete question and now it reveals that AOP basically complements OOP and cannot be used in isolation because AOP is developed on the basis of OOP. AOP counterbalances the constraint of OOP. When applied together with OOP, AOP is more efficient and complementary in providing an ideal structure for modular programming.

The scope of aspect-oriented implementation –that either it solves a specific cross cutting concern or it can be applied in general to the whole application – is to be well estimated by the metrics, so that to ensure the risks involved and opportunities offered by AOP. There are several factors that affect the performance of the application like main memory size, memory management, cache size and even program size (line of codes, etc.). Switching between the base code and the aspect is more often resulting in the back and forth movement of the control flows of the system, with the potential increase in the number of join points.

ACKNOWLEDGMENT

Many thanks to Saeeda Sultana Sadia, that we are able to complete the research. It could have been near to impossible to achieve the successful completion without the supervision and guidance of Saeed ur Rehman - Mentor. The cooperation of Department and faculty is heartedly acknowledged. Special thanks to Salma Sultana and Sara Sultana for their support and motivation to complete the research successfully.

REFERENCES

[1] R. J. Walker, E. L.A. Baniassad, and G. C. Murphy, "An Initial Assessment of Aspect-oriented Programming, ICSE99 Proceedings of the 21st international conference on Software engineering", ACM, ISBN: 1-58113-074-0, 1999, pp. 120-130.

[2] G. Kiczales, J. Lamping, and A. Mendhekar, "Aspect-oriented Programming, Proceeding of 11th European Conference of Object-Oriented Programming", LNCS 1241, 1997, pp. 220-242.

[3] S. K. Otrappa and P. J. Kulkarni, "Multilevel Security Using Aspect Oriented Programming AspectJ, Advances in Recent Technologies in Communication and Computing (ARTCom)", 2010 International Conference, IEEE, ISBN: 978-0-7695-4201-0, 2010, pp. 369 – 373.

[4] H. Li, M. Zhou, G. Xu, and L. Si, "Aspect-oriented Programming for MVC Framework, Biomedical Engineering and Computer Science (ICBECS)", 2010 International Conference, IEEE, ISBN 978-1-4244-5315-3, 2010, pp. 1 – 4.

[5] B. Amar, H. Leblanc, B. Coulette and C. Nebut, "Using Aspect-Oriented Programming to Trace Imperative Transformations, Enterprise Distributed Object Computing Conference (EDOC)", 2010 14th IEEE International, IEEE, ISBN 978-1-4244-7966-5, 2010, pp. 143 – 152.

[6] S. Hanenberg, S. Kleinschmager, and M. J. Walter, "Does Aspect-Oriented Programming Increase the Development Speed for Cross-cutting Code? An Empirical Study", ESEM '09 Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement, ACM, ISBN: 978-1-4244-4842-5, 2009, pp. 156-167.

[7] J. Zhang, and Y. C. G. Liu, "Modeling Aspect-Oriented Programming with UML Profile", 2009 First International Workshop on Education Technology and Computer Science, ISBN: 978-0-7695-3557-9, vol. 2, 2009, pp. 242-245.

[8] L. Madeyski, and L. Szała, "Impact of aspect-oriented programming on software development efficiency and design quality: an empirical study", Software, IET, IEEE, ISSN 1751-8806, 2007, pp. 180 – 187.

[9] D. Zhengyan, "Aspect Oriented Programming Technology and The Strategy Of Its Implementation, Intelligence Science and Information Engineering (ISIE)", 2011 International Conference, IEEE, ISBN 978-1-4577-0960-9, 2011, pp. 457 – 460.

[10] M. Bartsch and R. Harrison, "An exploratory study of the effect of aspect-oriented programming on maintainability", Software Quality, vol. 16, no 1, 2007, pp. 23-44.

[11] R. Coelho et al. , "Assessing the Impact of Aspects on Exception Flows: An Exploratory Study Proceedings of the European Conference on Object-Oriented Programming", 2008, pp. 207-234.

[12] P. Greenwood et al. , "On the Impact of Aspectual Decompositions on Design Stability: An Empirical Study", Proceedings of ECOOP 2007, pp. 176-200.

[13] K. Gybels and J. Brichau, "Arranging language features for more robust pattern-based cross-cuts", Proceedings of AOSD, 2003, pp. 60-69.

[14] M. Kuhlemann and C. Kästner, "Reducing the Complexity of AspectJ Mechanisms for Recurring Extensions, In Proceedings of the Second GPCE Workshop on Aspect-Oriented Product Line Engineering (AOPLE)", 2007, pp. 14-19.

[15] J. Zhang, Y. Chen, G. Liu, and H Li, "An Aspectual State Model and its Realization based on AOP, Proc. of WRI World Congress on Soft. Eng"., vol.3, 2007, pp. 163-166.

[16] T. Osogami and S. Kato, "Optimizing System Configurations Quickly by Guessing at the Performance, Proc. of ACM Special Interest Group on Measurement and Evaluation", 2007, pp. 145-156.

[17] C. A. Cunha, "Reusable Aspect-Oriented Implementations of Concurrency Control Patterns and Mechanisms, Proc. of the Aspect-Oriented Software Development", 2006, pp. 134 –145.

[18] W. Liu, C. Lung, and S. Ajila, "Impact of Aspect-Oriented Programming on Software Performance: A Case Study of Leader/Followers and Half-Sync/Half-Async Architectures", COMPSAC 2011, 2011, pp. 662-667.

[19] G. Kiczales, E. Hilsdale, and J. Hugunin, "An Overview of AspectJ", In Proc. ECOOP 2001, LNCS 2072, Berlin, June 2001,Springer-Verlag, 2001, pp. 327-353.

[20] R. Douence, T. Fritz, N. Loriant, J. M. Menaud, M. S. Devillechaise, and M. Suedhol, "An expressive aspect language for system applications with Arachne". 4th Int. Conf. on Aspect-Oriented Software Development (AOSD '05), Chicago, Illinois, Mar. 2005. ACM, pp. 27–38.

[21] F. Kato, K. Sakamoto, H. Washizaki, and Y. Fukazawa, "Comparative Evaluation of Programming Paradigm: Separation of Concerns with Object-, Aspect-, and Context-Oriented Programming," Proceedings of 24th International Conference on Software Engineering and Knowledge Engineering (SEKE 2013), pp. 594-599.

Design and Innovation in Game Development

Observations in 7 Small Organizations

Erno Vanhala, Jussi Kasurinen and Kari Smolander
Software Engineering and Information Management
Lappeenranta University of Technology
Lappeenranta, Finland
erno.vanhala | jussi.kasurinen | kari.smolander@lut.fi

***Abstract*—Design and innovation of game software is considered to be a creative task, which also involves methods from software development. But how do the game organizations actually design their products and innovate? The objective of this paper is to understand how game products are designed, what factors affect the design process and how game designers innovate. This study observed and analyzed seven game-developing organizations to allow comparison of their used design methods, design objectives and sources of their innovation. Based on our study, the game organizations regardless of their size are generally driven by the business factors, such as expected sales, in product design. Even though several organizations promote innovation and creative design, the business practicalities require the organization to prioritize to products that have high profit expectations. The findings indicate that the game development organizations acknowledge originality and creativity in their product design, but their major objective in the design work is to confirm marketability and business potential of the product.**

Keywords- Game design, innovation process, game industry, design restrictions

I. Introduction

Game development is a creative field of industry. Its software development tasks are also a means of expression [1], meaning that the development and design work is much more than just collecting and realizing the functionality and quality criteria for the new product. Unlike conventional software, game products do not have the requirement to fulfill a certain purpose and do it efficiently. Instead they are required to provide entertainment and keep the player interested in the product.

However, there are also studies on the game industry that see game development as comparable to normal software design and development [2, 3]. In some occasions, the promotion of creative chaos and informality may even be a publicity stunt to maintain an illusion that the game business is more relaxed or artistic, or at least less money-centric than conventional software development [1]. In the development of new products for popular, existing franchises this can be considered to be somewhat true, since there are established markets and a customer base for a certain type of product. However, in the development of new concepts, trends and franchises there still is room for innovation, since the game markets thrive for novelty factors and products, which offer something new to the user experience. This innovation and design for novel concepts is especially thriving in small and medium-sized game studios that are still searching for their first breakthrough product and trademark franchise [1].

In this paper, we study the innovation processes and design principles in small and medium sized game developing software organizations. The objective of this paper is to identify how game developers design their products, what factors affect the design in practice and what is the source of innovation in these organizations. Overall, the research questions were *"How game studios design their products"* and *"How game-developing organizations innovate and make business?"*. Our research group interviewed 27 professional game developers from seven game developing organizations to observe how game developers innovate and design game products. These 27 interviews were conducted with several stakeholders in the organizations, game designers, developers, project managers and upper management, to gain a comprehensive view into the game organizations and to understand how these organizations innovate and design in game development.

This paper is also related to our earlier studies on game developing organizations and innovation. In the earlier publications, game organizations have been studied from the viewpoints of technical infrastructure [4], organizational processes [5] and application of new technologies [6].

The rest of the paper is structured as follows: In Section 2, a number of related studies are introduced and assessed. In Section 3, the applied research methods are introduced and the results are presented in the Section 4. Section 5 discusses the study observations and Section 6 closes the paper with conclusions.

II. Related Research

Game business has been a growing area of industry for the last decade [7], regardless of the economic turbulences in other global business areas. This has driven up the number of game studios in many countries such as United States [7] or Finland [8], and increased the demand for new products and novel concepts.

Game design has been addressed in a number of publications. For example, a study by Blow [2] has identified the increasing complexity of game products during the last ten years. Due to increased processing power of the game platforms, the game products are able to simulate more sophisticated concepts, and at the same time allow more complex designs for new products. In addition of increased computing power, the game industry has also developed a fairly stable environment of well-known release platforms. The major shareholders, such as Sony or Microsoft are influential enough to form a de-facto industry standard [9].

Dymek [10] discusses the sources of innovation and the relationship between the software and game industry. The usual problem with the development models in the game industry is that the models overestimate the technology needs of game products, because the game industry is usually associated closely to the software industry. From the viewpoint of the game industry, games are cultural products that in the design process resemble more interactive movies than software [10]. However, Kanode and Haddad [3] have identified the most common problems in game development projects and point out that the most common problems are related to project management and development processes. The creative work is mostly used to develop the design for a game concept, and then later applied to refine the design “to find the fun”. Callelle et al. [11] agrees with Kanode and Haddad, mentioning that the development of a game design document is the most important design-phase work.

Kultima and Alha [1] identified seven profiles for people working in the games industry. The most common profiles were called “Instrumentalists” and “Artists”. The instrumentalists were people were able to identify useful or interesting characteristics in the applied platforms. The artists were the more common type of innovators; their drive to work in the game industry was based on the need to create something new. Interestingly, the third most common group was the “Nihilists”, who had a negative view on innovation. Almost every sixth interviewee was very critical towards innovativeness of the game industry, or innovation for the sake of innovation.

From the business viewpoint the game industry has gone through a paradigm shift from arcade video game halls to massive multiplayer online games and mobile games [12]. In games, new business and revenue models have been recently taken into use, including free-2-play or in-game advertisement models [13-17].

Computer gaming industry is also special in the sense that it can implement advertising embedded in games as value-adding parts [14]. Especially this is seen in sport games, where, for example, football players have real team outfits with sponsor tags on them. Gamers’ attitudes towards advertising is also more permissive than those of the people who do not play games [18]. This has made it possible, for example, to develop the free-2-play business model [19, 20], where games can include advertising and in-game purchasing can be done to monetize the game.

Traditionally in games, there has been a game package to buy, but currently digital distribution has started to eliminate this expense. Vanhatupa [21] claims that browser-based games can be offered for free and still get a steady long-term revenue stream by selling extra features and/or advertisements. This means that besides actual games, game companies always need to develop a working business model to monetize their ideas and technological innovations as technology itself has no value [22].

Overall, it seems that the game design is strongly related to the development of novel concepts and innovation for new ways to use the existing systems [2, 9]. The game industry sees itself more creative than “traditional” software industry, but in practice it seems that the most of the creative work is done when establishing new brands and franchises, and that the creative needs of game development are not that critical as expected [1,3,11]. On the business side, new technologies and business models cause further development needs for the ways how games are developed [19,21,22].

III. RESEARCH METHOD

The software process including the design, development and testing of a commercial product is a complex phenomenon, which has varying approaches even with seemingly similar organizations [23]. Acknowledging this, we decided to pursue empirical qualitative analysis by applying the grounded theory method [24-26]. We considered Grounded theory suitable for discovering and analyzing the activities done during a software project, as it observes and describes real-life phenomena within their social and organizational context. According to Hughes and Jones [27], the method suits well to these objectives.

Our approach is in accordance with the Strauss and Corbin [24] approach and in the process of building a theory from the case study research, we followed guidelines as described by Eisenhardt [28]. The interpretation of the field study results was completed in accordance with principles derived from [29] and [30].

A. Data Collection

The initial strategy for the population criteria and selection was based on our prior research experiences on conducting industry-wide studies on software industry in general, made by our research group [for example 23, 31]. We carried out four interview rounds in our study (Table 1) with four different interviewee groups; project managers, game developers, upper management and game designers. The sample of the interview rounds consisted of seven game development organizations selected from our research partners and supplemented with additional volunteering organizations to achieve a heterogeneous group of different target audiences, development platforms and organizational histories. Overall, 27 interview sessions were held during the spring, summer and fall of 2012 by seven researchers from two research laboratories.

The 7 organizations in the study group were small to medium-sized professional game companies. Five of the seven were either recent business startups or new companies (less than five published products) and two were more experienced organizations with more than five published titles. The selection of the cases was based on the polar type selection [28] to cover differences between organizations; the cases included different target platforms and different sizes of development projects. In practice, the organizations were selected from a number of volunteering research partners and supplemented with additional organizations. These organizations varied (Table 2) from newly started mobile game developers to browser-based games, PC games offered through digital distribution and even included an established developer with products in the retail stores. The smallest organization in the focus group was a startup with three persons; the largest organization included several hundred people that contributed to the product development. All of the participating organizations were commercial companies, with game development their main source of income.

The objective of this approach was to gain a broader understanding of the practice of and to identify the general factors that affect the design and innovation work. To achieve this, our research team developed four questionnaires that included questions on themes such as design methods, development processes, quality, business models and innovation. Before the first interview round the questionnaire was peer reviewed within the research group to check for sanity, and between the interview rounds some follow-up-questions were added to collect more details and test observations. All of the complete questionnaires are available at http://www2.it.lut.fi/project/SOCES/.

The interviews contained semi-structured questions, and the whole sessions were tape-recorded for qualitative analysis. Typically, an interview lasted for approximately one hour and they were arranged as face-to-face interviews with one or two organization participant and one or two researchers at the location selected by the interviewees. As we wanted to test and further flesh out our initial findings and observations from the earlier rounds, the interview rounds were conducted in order; for example the interviews with the second round interviewees started only after all first round interviews were conducted. Because of this and scheduling problems, we were unable to interview one representative during the second interview round, but the round-specific topics were discussed with the organization representatives on the latter interview rounds.

The decision to interview project managers during the first round was based on our aim to gain a better understanding of the operational level of software development. We wanted to see whether our observations and experiences from [23,31] the software industry were applicable in the game industry context.

The interviewees in the second round were selected from a group of developers or programmers, who directly contributed to the software product and had experience with the technical details of the developed product. To gain more insight into the technical infrastructure, the interview topics in this round were heavily focused towards programming techniques, process activities and applied development tools.

In the third round, the focus of the interviews was to collect more general data on the company beyond the development process of the products. During this round additional themes beyond the software development such as marketing, innovation and financing were collected to better understand the context in which the game industry operates. In the fourth round, the focus was on the creative aspects of the game development, in the design work. During this round the interviewed employees were game designers, or management-level personnel with the ability to affect the final design of the developed product.

The interview rounds, interviewee roles in the organization and study structure are summarized in Table 1, and the participating organizational units are summarized in Table 2.

B. Data Analysis

The grounded theory method contains three data analysis steps: open coding, where categories and their related codes are extracted from the data; axial coding, where connections between the categories and codes are identified; and selective coding, where the core category is identified and described [24].

The objective of the open coding was to classify the data into categories and identify leads in the data. The process started with "seed categories" [33] that contained essential stakeholders and known phenomena based on our prior studies in this context. Seaman [33] notes that the initial set of codes (seed categories) comes from the goals of the study, the research questions, and predefined variables of interest. In our case, the seed categories were derived and further developed from our prior studies on software industry. Our selection for the seed categories included general phases of the software processes such as design, development, testing and project management, and common terms and stakeholders such as financers, customers, project personnel, software tools and quality; areas and concepts which should exist in software development but which are not too restrictive or descriptive to bias the collected data. These seed categories were also used to define the themes for the questions in the questionnaire. The final data collection instrument, a series of open questions, included topics such as development process, test processes, tools, quality, design

TABLE I. INTERVIEW ROUNDS AND THEMES

Interviews	Interviewee	Description	Main themes of the interviews
Qualitative interview with 7 organizations	Team leader or project manager	The interviewee is responsible for the management of the development of one product, or one phase of development for all products.	Development process, test process, quality, outsourcing, development tools, organizational aspects.
Qualitative interview with 6 (+1*) organizations	Developer or tester	The interviewee was responsible for the development tasks, preferably also with the responsibilities of software testing activities.	Development process, test process, development tools, development methods, quality.
Qualitative interview with 7 organizations	Upper management or owner	The interviewee was from the upper management, or a business owner with an active role in the organization.	Organization, quality, marketing, innovation and design process, development process.
Qualitative interview with 7 organizations	Lead designer or Art designer	The interviewee was a game designer, or managerial level person with the ability to affect the product design and selection of the implement features.	Development process, design and innovation, testing, quality

* Interview themes discussed during later rounds with other representatives of the organization

process and finances, weighted between rounds based on the roles of the interviewees.

In open coding, the classified observations can be organized into larger categories. New categories appear and are merged because of new information that surfaces during the coding. For example, our initial concept of infrastructural problems being a seed category was abandoned as the coded interview data proved that the process problems were more related to personnel and management, technical issues having little to none observations in the study group. Similarly, several observations in different categories and issues which emerged from the data formed the coding for our data. Overall, at the end of the open coding, the number of codes was 172 codes with 1574 individual observations, collected from over 1400 minutes of recordings from 27 interview sessions.

The objective of the axial coding, which starts when the categories start to emerge and runs somewhat parallel with the open coding [24], is to further develop the categories by looking for causal conditions or any kind of connections between the categories. In this phase, the categories and their related observations were becoming fixed, allowing the analysis to focus on developing the relationships between larger concepts. In this phase, the categories formed groups in the sense that similar observations were connected to each other. For example, codes such as "Design process: refining designs", "Development process: knowledge transfer" and "Problem: Documentation/knowledge transfer related to design" formed a chain of evidence of how the organization documented and refined their product designs and what problems the designers and developers had with this approach. By following these types of leads in the data, the connections between categories were identified and made.

The third phase of grounded analysis, selective coding, is used to identify the core category [24] and relate it systematically to the other categories. The core category is sometimes one of the existing categories, and at other times no single category is broad or influential enough to cover the central phenomenon. In this study, the examination of the core category resulted to the category *"Overall Objectives of the Innovation and Design in Games"*, which is an umbrella category explaining the observations related to design work, innovation and long-term objectives the organizations have.

The core category was formed by abstracting the categories and most important issues as none of the existing categories was considered influential enough to explain the entire phenomena. For example, we observed that the primary method of design work was based on one individual, who made the decisions based on group work, and that in all organizations the objective of the development work was in economic aspects, not in artistic presentation or other non-economic issue even though these topics were discussed in some organizations. In addition, the most important limitation was resources, specifically time, not the release platform or available tools. Additionally, we also observed that the most important source of innovation was previous experience with game products, and somewhat surprisingly the other cultural sources such as folklore or literature were not used to a large degree. We adjusted the core category *"Overall Objectives of the Innovation and Design in Games"* to include all of the categories and observations, which discuss the objectives of the design work in organizations before the actual development starts, the sources of innovation in the organization and the overall effect the marketing and financial aspects have on the game product design work.

TABLE II. DESCRIPTION OF THE ORGANIZATIONS

	Release platforms	**Production team size**[1]	**Maturity. amount of released games**
Case A	PC, game consoles	Large	Established, more than 10 released products
Case B	Mobile platforms	Small	Recent startup, Less than 5 released products
Case C	Game consoles, PC	Large	Established, Less than 10 released products.
Case D	Mobile platforms, PC	Medium	Startup, developing first product
Case E	Mobile platforms	Small	Recent startup, less than 5 released products
Case F	PC	Medium	Startup, developing first product
Case G	Browser games	Small	Startup, developing first product

[1]Amount of people contributing to the released product, size by SME

IV. RESULTS

In this section we discuss the analysis results. The categorized observations and main findings are presented in Table 3, and the connections between the categories in Figure 1. After explaining the main categories we introduce the findings on game design methods and innovation and the effect of business aspects on the game design. Finally, we discuss the implications of the results.

A. *Categories*

The core category, *Overall Objectives of the Innovation and Design in Games*, is a composition of several categories, which all discuss the design work, innovation or aspects that affect the design work or innovation. The categories were formed inductively from the interviews. They explain the relationship between the design objectives and innovation process, or the effects of business practices affecting the product-related decisions. These selected categories describe how our case organizations approached design process and how business factors affected the product design.

The category *Objectives of the design phase* summarizes the most important objective the organization has for the design work. In most organizations the objective was on exploring the game concepts and testing that the potential new product could be marketable, fun to play and with proof-of-concept prototypes, doable with the target platform.

The category *Design method* describes how the organization designs their new products. *Vision* means that the organization has lead game designers that draft the first concept based on their own ideas. *Idea pitching* means that the organization applies open sessions where employees can pitch their ideas, and the most liked ideas are further studied.

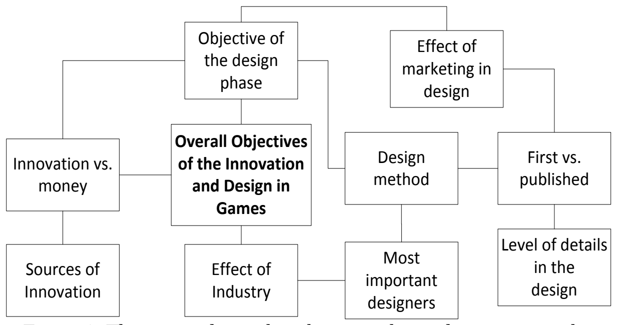

Objective of the design phase
Effect of marketing in design
Innovation vs. money
Overall Objectives of the Innovation and Design in Games
Design method
First vs. published
Sources of Innovation
Effect of Industry
Most important designers
Level of details in the design

Figure 1: The main relationships between the study categories; the lines represent categories which share related features.

Brainstorming means that the development team organizes dedicated design sessions, in which they make the first designs for potential new products as a group effort. *Prototypes* mean that the organization develops crude prototypes to explore their new concepts and decide which prototype to develop to a full game based on their look and feel. *Pen and paper* means that the organization has designers or artists, which create mock screenshots and concept drawings to flesh out concepts which may be based on personal ideas or a group effort.

The category *First vs. published* product indicates the amount of differences between the typical first functional prototype of a game product and the final outcome. *Major changes* indicate that the game may have large changes in the design, including genre, theme, release platform or main marketing features. *Minor changes* indicate that the changes are only related to the smaller features, such as amount and type of game content, game mechanics, changes in creative writing or control scheme. In Case G this category was divided to technical and game design, since their game had only minor changes content-wise, but underwent drastic changes in the technical solution.

The category *Level of details in the design* describes the amount of details in the initial design, which is used to start the development of an actual product. *Functional prototype* indicates that the organization develops a proof-of-concept prototype, which has all of the intended main features of the game to assess the feasibility of the product design. If the design is considered usable and marketable, then the development team starts to build an actual product. *Basic gameplay elements* mean that the organization designs a functional concept with the basic features, story elements, themes and characters with some technical studies on concept feasibility. *Core features and concept art* is one step towards simple draft documentation; the main features and some concepts for theme and creative aspects are drafted but usually no programming work is done.

The category *Effect of industry* describes the ways the organization considers the games industry in general to affect their product design, marketing approach or business models. Case organizations A, B, C, D and F considered the industry to affect mostly on the required features of the game; customers expect some abilities such as hand gestures or platform-specific functionalities which demand the designers to cater to these expectations. Cases C, E and G also mentioned that the industry affects their business model, either by forcing the organization to constantly update their products (Case C) or by opening new market segments or revenue models such as free-2-play [20].

The category *Most important designers* indicate in the project-level who in the case organization actually leads the design work for new product. *Producer* indicates that in the organization the design decisions are ultimately made by the project manager, who supervises the designers, developers and game artists. *Lead designer* means that the organization has a separate role for the person who makes the decisions on designs and can dictate what features are included and

TABLE III. OBSERVATIONS FROM THE CASE ORGANIZATIONS AND CATEGORIES RELATED TO THE FINDINGS

	Case A	**Case B**	**Case C**	**Case D**	**Case E**	**Case F**	**Case G**
Objective of the design phase	Make something that sells, marketable in near future	Concept demo on technology, game mechanics	Test if the concept is fun	Good mechanics, game that sells	Test mechanics for concept, something that is fun.	Design of own thing, things selling are old six months	Design something we are very good at making
Design method	Idea pitching, prototypes, brainstorming	Vision, brainstorming	Vision, Idea pitching, prototypes	Vision, pen and paper	Brainstorming, prototypes, pen and paper	Prototyping, Vision	Vision
First vs. published	Major changes	Minor changes	Major changes	Major changes	Minor changes	Large major changes	Large technical, minor design
Level of details in the design	Functional prototype	Basic gameplay elements	Functional prototype	Core features, concept art	Basic gameplay elements	Core features, concept art	Basic gameplay elements
Effect of industry	Enforces features	Publisher sets requirements	Enforces upkeep, adding new content	Changes to design	New customers, business models	Stabilizing effect on designs	"Marketing dictates success"
Most important designers	Producer	Lead designer, team	Producer	Lead designer	Team	Management	Lead designer
Innovation vs. money	Money first, then innovation	Money first, then innovation	Innovation, hopefully money	Money first, then innovation	Money first	Money first (free2play)	Money first, then innovation
Effect of marketing in design	"We design fun, management handles sales"	"Has to be profitable"	"Make fun demo and sell it"	"Business first"	"Business first"	"Good game sells"	"Finances has to be taken into account"
Sources of innovation	Movies, other games	Success stories, industry trends	Success stories	Prior experiences, old games	Platform possibilities, old games	Movies, books, TV, games, "portfolio of stuff"	Prior experiences, competition analysis

excluded from the product. *Team* indicates that the decisions on game design are made by the entire development team, with more or less democratic system of discussions and voting. *Management* indicates that the design is directly overseen by the management above the development team, and deviations from the original design have to be accepted by them.

The category *Innovation vs. money* describes whether organization units are aiming to build financially successful business or are motivated by developing their creative idea into a product and "hoping" it can produce income. All the companies, except Case C, are going with the philosophy money first, where they first build products that generate profit and after that start building their dream products.

The category *Effect of marketing in design* describes how the marketing aspects affect the game design. Cases A, C and F considered the design work to be separated from marketing, indicating that the most important objective of design work is to come up with a creative and fun concept, with management or marketing focusing on how to sell that design. In other case organizations the design starts with a market study on what could be a financially feasible product, and based on the market study the product is designed and developed so that it fits the target audience.

Finally, the category *Sources of innovation* describes the main sources of innovation and ideas for the designers. Cases A, B, C, D and G named the other, earlier success stories of the games industry as one of their most important sources of innovation, meaning that the organization did markets studies such as "what sort of games sell" and "why did this game become success". Other usual sources for innovation and ideas were prior gaming experiences and old games in general.

B. On design process, design objectives, innovation and business

The organizations shared two common features in the design work. First, all organizations based their design work on economic issues, placing financial success over critical success. In other way, all organizations expressed that should they choose between highly innovative and memorable but financially adequate and financially successful but forgettable product, they would aim for the financial success. Secondly, all organizations considered that the available resources, mostly time, was their most limiting design factor. As the case organizations had to plan their product publications within a foreseeable timeframe – usually 3-12 months –, in all organizations the design, development and testing tasks did not have much excess time to fine-tune the technical implementation or user experience beyond an acceptable level of quality.

"... after all, there really is very limited amount of time to do surprisingly large amount of tasks." – Case B, Lead Designer

"*I don't think that there really are [technical] restrictions to creativity, it's just that there are limited amount of people." and " ..."too few people, too little time, too little money.*" – Case E, Lead Designer

Besides these two observations, our analysis also yielded six main findings describing how the game organizations do design and innovation work. In following, we will introduce these findings one by one.

1) Game product design is driven by economic factors.

In most organizations the game design is strongly related to the financial potential of the game product. Even if the game industry in general is seen as a creative industry, the product design follows mostly economic principles. In all organizations with the exception of Case C, the organization considered the profits to be more important than innovation.

"*It is nice if the critics and people like your game, or if it is a review hit, but it may not translate into profits. If I had to select between [money and publicity] I would definitely go with money."* – Case E, Project manager

"I would like to make a game that has cultural impact, or at least is very well known for artistic merits. However, first we need to have significant financial successes…" – Case D, Upper management

In most organizations the tradeoff between innovative and money-making products was that the organization needed money first to build innovative, experimental products later. This approach also affected the design objectives. In cases A, B, D, E and F the organization was designing their products based on the marketing potential or business-first approach. In case C and F the organizations were geared towards more innovative design. These organizations considered that well-made games sell themselves, so a good design makes a game easy to sell. Case A expressed similar sentiments, but ultimately held financial potential as the most important design objective.

"*Our strategy is based on our analysis on what is going on, what are the most potential, growing areas, and where it is most likely to get our investment to resources back."* – Case A, Project manager

Cases F and G had additional considerations for their product design. In Case G, the product design was examined with proof-of-concept prototypes to ensure that the product was possible to develop for the target platform. In Case F the design focused heavily into doing "own thing". As it takes at least six months to develop a game, any product resembling the themes and concepts of the current top-selling products would be "old news" and a past trend when released.

"If we look into the best seller list of [platform] right now, they probably no longer sell in six months."..."When our game after months and months of development is released, it is nothing new or exiting. That is why we should do something different." – Case F, Lead designer

2) Design relies on prototypes, which test out potential game concepts

Game organizations heavily rely in the prototyping approaches in their designs. In Cases A, C, E and F the organization did design work by studying the game concept with varying degrees of prototypes. This approach was applied to ensure that the created design also worked in the actual implementation.

"We make a prototype to test if the concept is actually fun to play with and ensure that it has the needed potential." – Case C, Project manager

The two organizations that had already released a number of games, built functional prototypes as the first design version (Cases A and C). The organizations that were building their first product relied merely on concept art and a list of core features (Cases D and F). This may indicate that early start-ups do not yet have the skill to build a working prototype, and therefore they focused on concept art only.

"We started by simply thinking what sort of control mechanics are used in mobile games, based a simple design on top of that and with pen and paper, tested, thought out and developed a first build." – Case E, Project manager

3) Most game designs are based on a concept innovated by individuals

The design work in the development of new products was heavily focused on one or few individuals in the organization. In Cases B, C, D, F and G the first concept of a new game product came from a designer, or a person who came up with an idea that was feasible to implement. After the initial idea, Cases B and C worked in teams to flesh out the idea, whereas in Cases D, F and G the design was still in hands of one or few individuals.

"I am responsible for [making design decisions]. I have to do the final call, since groups simply do not sometimes have that ability." – Case B, Lead Designer

"I make the decisions, but usually based on the group input" – Case D, Upper Management

In Cases A and C the design work started with an idea pitching event, where each individual could propose new ideas for new products. Case A was more geared towards making a communal decision within a group to select the best concepts, whereas Case C relied more on the work of the individuals to convince the group to their game concept.

"When someone gets an idea, they can show their ideas on these concept cups."…"If enough people like it we take it forward to design." – Case C, Developer

In all organizations with the exception of Case E and – to a lesser extent Case F – the product design and decisions on included and excluded features was the responsibility of one named person. In Cases B, D and G this person was a lead designer, who in all cases was also the person responsible for making the first design. In Cases A and C the design changes were managed by the game producer, a project manager, who made the decisions on what the product should include and exclude.

"We sit down and have a team discussion once in a fortnight to see where we are and discuss new ideas. After these sessions the producer goes through the ideas and what can be included and what not, and includes feasible stuff to the next sprint." – Case A, Upper Management

The Cases F and B are exceptions to the strong creative control observed in other studied organizations. In Case F the upper management had a direct control over the aspects of the developed games. In this organization the creative control was outside the development team. However, the upper management was also responsible for designing new products for the organization. In Case E the design work and change management was done as a group effort. The design was changed only if everyone or at least most of the development team approved the idea. The first idea was developed in brainstorming sessions, explored with prototypes and fleshed out as a group effort. Unlike Case B, which had similar activities in the design (pre-production) phase, Case E did not have a separate lead designer or decision maker for creative aspects at any stage.

"With our first game, we really did not have specific planning phase, we simply went as a group and decided to do something simple, something like a proof of concept for our team being able to make games." – Case E, Project manager

"We just brainstorm within our development team, there really is no further magic to [design work]." – Case E, Upper Management

The most important designer in the project was also related to the age of the company. Cases A and C had been in the business longer and they reported that their most important designer is the producer, whereas the smaller and newer companies did not report that such a person even existed. This is a bit similar as with functional prototypes in finding 2. The early start-ups had not yet grown big enough to have their own producers.

4) Design and innovation are ad-hoc processes

The Cases report various design and innovation methods, like idea pitching, brainstorming, group work and pen and paper. Yet, none of the cases report that they have used more formalized ways of design, like lateral thinking [34,35] which can be used also as a tool to build completely new ideas. Although brainstorming can be considered as a more formal method [35,36], its whole potential was not used by the organizations as interviewees did not explain any systematic use of the method.

"Personally my ideas are born when I have slept overnight and I am driving a car by myself and I have some time to think." – Case G, Upper management

The companies relied more on ad-hoc innovation, which could be because they were not aware of the more formal methods. As for these methods, brainstorming and idea pitching can be seen as semi-formal methods. In idea pitching the new idea has to be presented with maximum of three slides and after that decision is made whether functional prototype is build or not.

Cases A, B and C mentioned "game concept day" or "proto day" as a day when developers discuss and develop new concepts and prototypes. This can also be seen as semi-formal method as the aim is to produce new ideas.

"If these ideas are developed further, there is reward given." – Case A, Upper management

One interviewee mentioned a reward system as a motivational factor in the innovation process. Its usefulness is unclear, but Case A had been in the business for some time, this system seems to work at least to some degree.

5) Sources of innovation are mostly in existing game products and success stories

The most important sources for innovation and ideas for new products were old games released for older generation of game systems and popular, successful game products of the current markets. All interviewed game designers indicated that they used their past experiences with game systems and old games as one of their source of innovation.

"Our newest game is inspired by this old game from the 90's… it basically was the initial model for our design. We made our thing on top of that." – Case D, Lead Designer

Beyond prior experiences with games, some of the case organizations did actual market reviews and analyzed success stories. In Cases B, C, F and G the organization paid close attention to the business, analyzing why some games were successful and what sort of features the current successes had incorporated. Case E added also technical point of view into these analyses.

"We know about markets enough because we took our demo to [industry convention] and talked with people. We met over 30 people from the industry to understand what publishers look for"…"Now we know that we are doing the right thing." – Case G, Upper Management

"With our prototypes we also test out to see if the technical solution is capable of doing what we want it to do." – Case E, Project Manager

Besides success stories, existing products and competition analysis, other sources for innovation in product design were movies, books and other popular media. The only popular media that was mentioned several times as a source of innovation was summer blockbuster movies.

"…Also movies, we use movie references really too much." – Case A, Lead Designer

6) Start-ups are business-driven in game industry

Six out of seven case organizations described their ideology as "money first" (see Table 3). We can argue that these companies have understood that technology itself has no value [22], as it is the responsibility of the company to monetize the technology. In addition four out of these six "money first" organizations described their marketing/finance design as "has to be profitable", "business first" or "finance has to be taken into account". The one organization that had the philosophy of doing "innovation, hopefully money" wanted to "make fun demo" and then sell it. With these opposite philosophies we saw that money played the most important role for almost all cases.

In addition to the rows innovation vs. money and effects of marketing in design, money and selling are also listed in three cases in objectives in the design phase. Although this paper focuses on design and innovation we also observed that selling, business and money were important issues for almost all the companies. For example, Case D goes with "money first", "business first" and its design objective is "game that sells"; they are going with business-driven development where the aim of software development is satisfy business requirements [37]. Case C, as an opposite, goes with "innovation", "make fun demo and sell it" and its design objective is to "test if the concept is fun". Although Case C has a different attitude than the rest of the organizations, it has still managed to establish itself.

In Figure 2 we present seven case organization units and both their number of released products and their business-drivenness. The latter is calculated from Table 3 by using rows objectives in the design phase, innovation vs. money and effects of marketing in design. If business/money is mentioned as a first thing 1 point is gained. If it is mentioned as second thing 0.5 points are gained. If it is not mentioned, no points are gained. Maximum is three points.

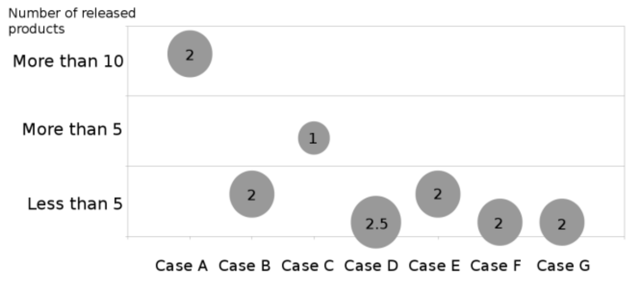


Figure 2: Number of released products from Cases and their business-drivenness

The Cases D, F and G are all making their first product and they are also business-driven as the lowest score among them is 2. On the other hand the rest of the companies have already released at least one game and among them the highest score is 2. As several cases described that they first aim to make profit and after that produce games they really want to do. Our observations support the concept that newly established game companies are more business-driven and think more about money whereas companies who have already released successful products can concentrate more on other than immediate economic issues.

"I would like to make a game that is a landmark… But first I aim that we can do economic success, which would give us economic freedom which would give us freedom to ourselves to do artistic game." – Case D, project manager

V. DISCUSSION

In this work the core category is the *Overall Objectives of the Innovation and Design in Games*. Based on our observations, the *game products are designed with creative processes comparable to movies or any other artistic creation, but games are not intended to be art for art's sake, they are designed and intended to be commercial products which generate income.* All game developers interviewed in this study considered themselves to be doing more or less creative work, but in all organizations the most important objective in product design was in commercial success.

The concept that games are designed based on business aspects can also be observed from the viewpoint of design principles. In some organizations the most important design aspect was in developing "fun" product, but in the long run the organization was still aiming at commercial success. When faced with the dilemma of selecting between a commercially successful but forgettable and critically acclaimed but commercially adequate product, all interviewees selected the commercially successful product. In all organizations marketing and marketability had at least some effects on the product design. In Cases B, D, E, F and G the financial aspects dictated the products the organization was developing, and even in the larger Cases A and C, the product had to have a clear audience and a reasonable expectation for profit before the product would advance from a proof-of-concept prototype onwards.

Considering the research questions, *"How game studios design their products"* and *"How game-developing organizations innovate and make business?"*, the results indicate that the design process is usually led by one individual, who uses the team input as suggestions. The initial concepts are heavily influenced by the "vision" of the new product, and the decisions on which designs mature from proof-of-concept prototypes to fully developed products is usually dictated by the potential for revenue. The common source for innovation in game development seems to be legacy games, experiences gathered from other game products and movies. The marketing and business aspects also heavily affect the innovation process.

None of the organizations used formalized methods when developing new ideas and concepts. The methods used were merely ad-hoc and ideas "just emerged" rather than were systematically developed, with a few exceptions of "proto days" and team brainstorming. In addition, companies seem to be more business-driven when they are starting up and establishing their position. After that they can be more innovative and concentrate less on monetizing ideas.

In grounded theory study, there are threats to validity. As the method of data collection was based on semi-structured interviews, threats such as personal bias caused by the researchers or questionnaire are valid concerns. For example, a study by Whittemore et al. [38] lists integrity, authenticity, credibility and criticality as primary criteria for validity in qualitative studies. The aim is to describe the observed phenomenon and the applied approach with enough details to warrant that the analysis process has been critically designed, unbiased and faithful to the data. Similar considerations have been expressed by Morse et al. [39]. The nature of the qualitative studies requires the presentation to constantly verify the collected data and analysis results to achieve the necessary rigor for a trustworthy qualitative study.

In our study, the validity concerns have been addressed with several precautions. The data collection instruments were developed by seven researchers from two different research groups. Before the first interview round, the data collection instrument was peer-reviewed for sanity and neutrality within the research group. The instruments were further developed during the data collection, and the data collection itself was conducted by six researchers. For this study, the data analysis was conducted and discussed by three researchers, with conflicts resolved with discussions during meetings. To minimize the bias caused by the release platforms, business types or interviewee roles, the interviews were collected from different types of interviewees, and the case study organizations were selected to represent different areas of game industry in business maturities, sizes and business platforms. In any case, these qualitative results are valid only in this environment, and beyond the scope of this study these results should be used as recommendations or indications of possible organizational activities.

VI. Conclusions

We have introduced our grounded theory study on the game developing organizations. We observed seven game developing organizations by interviewing 27 industry professionals encompassing different roles such as project managers, developers and game designers. Our results suggest that game design and innovation are closely related to the economic aspects of the game industry. The design objective is to generate income with development projects that are considered feasible for economic success. In many organizations the creative game design work is done by one person or a small group of people who have creative control over the project, although in some cases group decisions also have influence. The main sources of innovation in game design seem to be in the existing game products and industry success stories, with some novel concepts taken from popular media, mostly from movies.

The organizations in our study had different attitudes towards business and innovations. Whereas most of the organizations wanted to build their business on a business-driven model, one organization pushed successfully ahead with creativity, innovation and fun. It seems that start-up organizations are business-driven in the beginning because they need to established their position and secure their future in the industry.

The results of this study can be used to understand the business practices and development processes of the game industry. In future work, the business modeling methods and effects of marketing to the development processes should be addressed in more detail to study how much influence the business decisions have on the development in practice.

Acknowledgment

This study was supported by the European Union Regional Development Fund project number A31814, "Kaakon Peliklusteri", administered by the Council of Southern Karelia, Finland and the organizations funding the related research project. We would also like to thank all the interviewed organizations and the project partners, especially Cursor Oy.

References

[1] A. Kultima and K. Alha, "Hopefully Everything I'm Doing Has to Do with Innovation: Games industry professionals on innovation in 2009", Proc. 2nd International IEEE Consumer Electronics Society's Games Innovation Conference, Hong Kong, China, 2010.

[2] J., Blow, "Game Development: Harder Than You Think", Queue, Vol. 1(10), February 2004, pp. 28–37.

[3] C.M. Kanode and H.M. Haddad, "Software Engineering Challenges in Game Development", Proc. 2009 Sixth International Conference on Information Technology: New Generations, 27.-29.4., Las Vegas, USA, 2009. DOI: 10.1109/ITNG.2009.74

[4] J. Kasurinen, J-P Strandén and K. Smolander, "What do game developers expect from development and design tools?", In Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering (EASE '13). ACM, New York, NY, USA, 2013. pp. 36-41. DOI=10.1145/2460999.2461004

[5] J. Kasurinen, R. Laine and K. Smolander, "How applicable is ISO/IEC 29110 in Game Software Development?", accepted to the Proc. 14th Int. Conf. on Product-Focused Software Development and Process Improvement (Profes 2013), 12.6.-14.6.2013 Paphos, Cyprus, 2013.

[6] L. Riungu-Kalliosaari, J. Kasurinen and K. Smolander, ”Cloud Services and Cloud Gaming in Game Development”, accepted to the Proc. Cloud services and cloud gaming in game development, accepted to the Proc. IADIS International Conference: Game and Entertainment Technologies 2013, 22.-24.7. Prague, Czech Republic.

[7] Entertainment Software Association (ESA), “2011 Sales, demographic and usage data: Essential facts about computer and video game industry”, 2011.

[8] M. Peltoniemi, “Life-cycle of the Games Industry The Specificities of Creative Industries”, Proceedings of the Mindtrek’08, 7.-9.10.2008, Tampere, Finland.

[9] S. Gallagher and S.H. Park, Innovation and competition in standard-based industries: a historical analysis of the US home video game market, Engineering Management, IEEE Transactions on vol.49, no.1, Feb 2002, pp.67-82. doi: 10.1109/17.985749

[10] M. Dymek, Content Strategies of the Future: Between Games and Stories – Crossroads for the Video Game Industry, Proc. 3rd Int. Conf. on Digital Interactive Media in Enterntainment and Arts (DIMEA’08), 10.9.-12.9.2008, Athens, Greece, 2008.

[11] D. Callele, E. Neufeld and K. Schneider, “Requirements engineering and the creative process in the video game industry”, in Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on, pp. 240 – 250.

[12] P. Zackariasson and T.L. Wilson, “Paradigm shifts in the video game industry”, Competitiveness Review: An International Business Journal incorporating Journal of Global Competitiveness 20, 2010, pp. 139–151.

[13] T.R. Alves and L. Roque, “Because Players Pay: The Business Model Influence on MMOG Design”, Situated Play, Presented at the DiGRA 2007, Tokyo.

[14] M.R. Nelson, H. Keum, and R.A. Yaros, Advertainment or Adcreep? Game Players’ Attitudes toward Advertising and Product Placements in Computer Games. Journal of Interactive Advertising 5, 2004, pp. 3–21.

[15] A. Ojala and P. Tyrvainen, “Developing Cloud Business Models: A Case Study on Cloud Gaming”, IEEE Software 28, 2011, pp. 42–47.

[16] D.J. Teece, “Business Models, Business Strategy and Innovation”, Long Range Planning 43, 2010, pp. 172–194

[17] M. Yang, D. Roskos-Ewoldsen, L. Dinu and L. Arpan, “The Effectiveness of “in-Game” Advertising: Comparing College Students’ Explicit and Implicit Memory for Brand Names”, Journal of Advertising 35, 2006, pp. 143–152.

[18] S. Youn, M. Lee and K.O. Doyle, “Lifestyles of Online Gamers: A Psychographic Approach”, Journal of Inter 3, 2003.

[19] J. Hamari And A. Järvinen, A., ”Building Customer Relationship through Game Mechanics in Social Games. Business, Technological and Social Dimensions of Co mputer Games: Multidisciplinary Developments”, IGI Global, 2011, Hersey, PA, USA.

[20] H. Tyni, O. Sotamaa and S. Toivonen, S., “Howdy pardner!: on free-to-play, sociability and rhythm design in FrontierVille”, Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments. ACM, Tampere, Finland, 2011, pp. 22–29.

[21] J.-M. Vanhatupa, ”Business model of long-term browser-based games - Income without game packages”, 7th International Conference on Next Generation Web Services Practices (NWeSP), 2011, pp. 369–372.

[22] H. Chesbrough, “Business model innovation: it’s not just about technology anymore”, Strategy & Leadership 35, 2007, pp. 12–17.

[23] J. Kasurinen, O. Taipale and K. Smolander, K., “Analysis of Problems in Testing Practices”, Proc.of the 16th Asia-Pacific Software Engineering Conference, 1.12.-3.12.2009, Penang, Malaysia, 2009. doi: /10.1109/APSEC.2009.17

[24] Strauss, A. and Corbin J. (1990). Basics of Qualitative Research: Grounded Theory Procedures and Techniques. SAGE Publications, Newbury Park, CA, USA.

[25] B. Glaser and A.L. Strauss, “The Discovery of Grounded Theory: Strategies for Qualitative Research”, Chicago: Aldine, 1967.

[26] J.C. van Niekerk and J.D. Roode, J.D., “Glaserian and Straussian grounded theory: similar or completely different?”, Proc. of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, Vanderbijlpark, South Africa, 2009. DOI: 10.1145/1632149.1632163,

[27] J. Hughes and S. Jones, "Reflections on the Use of Grounded Theory in Interpretive Information Systems Research" . ECIS 2003 Proceedings. Paper 62. http://aisel.aisnet.org/ecis2003/62

[28] K.M. Eisenhardt, 'Building Theories from Case Study Research', Academy of Management Review, vol. 14, no. 4, 1989, pp. 532-550.

[29] G. Paré and J.J. Elam, “Using Case Study Research to Build Theories of IT Implementation”, The IFIP TC8 WG International Conference on Information Systems and Qualitative Research, Philadelphia, USA. Chapman & Hall, 1997.

[30] H.K. Klein and M.D. Myers, "A set of principles for conducting and evaluating interpretive field studies in information systems”, MIS Quarterly, vol. 23, 1999, pp. 67-94.

[31] J. Kasurinen, O. Taipale, and K. Smolander, “Software Test Automation in Practice: Empirical Observations”, Advances in Software Engineering, Special Issue on Software Test Automation, Hindawi Publishing Co., 2010. doi: 10.1155/2010/620836

[32] European Comission, "The new SME Definition User guide and model declaration" Enterprise and Industry Publications, European Commission, 2003.

[33] C.B. Seaman, "Qualitative methods in empirical studies of software engineering", IEEE Transactions on Software Engineering, vol. 25, 1999, pp. 557-572.

[34] D.J. Hall, “The role of creativity within best practice manufacturing”, Technovation 16, 1996, pp. 115–121.

[35] A. Kultima and J. Paavilainen, ”Creativity techniques in game design”, Presented at the FuturePlay 2007, ACM Press, Toronto, Canada, 2007, pp. 243.

[36] P.C. Shih, G. Venolia and G.M. Olson, “Brainstorming under constraints: why software developers brainstorm in groups”, Proceedings of the 25th BCS Conference on Human-Computer Interaction. British Computer Society, Newcastle-upon-Tyne, United Kingdom, 2011, pp. 74–83.

[37] J. Koehler, R. Hauser, J. Küster, K. Ryndina, K., J. Vanhatalo and M. Wahler, ”The Role of Visual Modeling and Model Transformations in Business-driven Development”, GT-VMT 2006. Presented at the 5th International Workshop on Graph Transformation and Visual Modeling Techniques, Vienna, Austria, 2006, pp. 1–12.

[38] R. Whittemore, S.K. Chase, C.L. Mandle, “Validity in Qualitative Research”, Qual Health Res, July 2001, Vol. 11, pp. 522-537. doi:10.1177/104973201129119299

[39] J.M. Morse, M. Barrett, M. Mayan, K. Olson and J. Spiers, “Verification Strategies for Establishing Reliability and Validity in Qualitative Reseach”, International Journal of Qualitative Methods, Vol 1(2)., 2002.

An Automatic Petri-net Generator for Modeling Multi-agent Systems

Meriem Taibi, Malika Ioualalen, Riad Abdmeziem
LSI - USTHB
Algiers, Algeria
emails: {taibi,ioualalen,abdmeziem}@lsi-usthb.dz

***Abstract*— A multi-agent system can be studied as a concurrent, asynchronous, stochastic and distributed computer system. These characteristics of multi-agent systems make them also a discrete-event dynamic system; it is, therefore, important to analyze the behavior of such system to ensure that it terminates correctly and satisfies other important properties. Several analytical methodologies have been used to study multi-agent system, particularly Petri nets. Petri nets have a well-defined mathematical structure that can be leveraged to provide formal analysis on discrete-event systems. In this work, we propose an automatic transformation to model multi-agent systems using Colored Petri nets.**

Keywords-Multi-agent system; Colored Petri net; Modeling; Description language.

I. Introduction

Multi-agent systems have been widely studied in the past few decades, where several frameworks have been defined in order to apply the multi-agent system concept to different applications in control and optimization of complex systems [1][2]. An agent is a computer system or computer program that presents several complex characteristics. A Multi-Agent System (MAS) [3] consists of a set of agents, interacting to achieve a common goal. Generally, MAS are known to work properly in a dynamic large-scale complex environment (open environment), thanks to several properties like: autonomy, adaptability, robustness and flexibility. The complexity and capabilities of a multi-agent system are greater than those presented in distributed software systems. In both cases, the study of system properties is becoming more important due to the fact that we are faced more and more to deal with large complex dynamic systems.

Tests and simulations have contributed for a long time to validate such systems. However, these techniques allow to investigate just a part of the global behavior. By that, they differ from the formal verification techniques, which ensure that a property is verified by all possible system executions.

Therefore, the important challenge in this field is the development of analytical methods to assess key properties of such systems. Such methods could be used to impart a preliminary analysis of the multi-agent system, providing design and operation feedback before the development of expensive systems. Many models based on $Belief - Desire - Intention$ (BDI) architecture were proposed in [4] and [5].

Other works include various attempts to deliver a formal model from AUML (Agent Unified Modelling Language) diagrams [6][7]. The advantage of these methods is that most developers are familiar with the (A)UML and an automatic transformation of their diagrams into formal models and model-check them, would greatly simplify the software quality control. The difficulty is that AUML diagrams allow much more freedom for the designer than formal models and the automatic translation is not trivial.

Petri nets have a well-defined mathematical structure that can be leveraged to provide formal analysis on discrete-event systems. In addition, Petri nets have been successfully used in several areas for the modeling and analysis of distributed systems [8].

Several studies have been proposed to model MAS with Petri nets (PN). In [9], a model was proposed for a promotional game of viral marketing on the Internet. Specifically, authors used stochastic Petri nets for modeling a multi-agent wish list. As well, Gazdare [10] used Colored Petri nets (CPN) as a formal method to model a transport system with containers, then, simulate and solve the storage problem. In EL Fallah-Seghrouchni [11], Boukredera [12] and Khosravifar [13], authors also proposed to use the CPN formalism to model interaction protocols.

In this work, we propose an automatic transformation for modeling multi-agent systems. This automation is based on two steps: first, the system is described using a language called MASDL, then, a set of transformation rules are applied to obtain the CPN models.

This document is organized as follows. Motivations and the problem statement are presented next. Section III gives an introduction to MAS. Section IV presents the main aspects of the language we define to specify MAS. In Section V, we present our transformation algorithm allowing to model MAS using CPN. Section VI presents our application, and finally, Section VII discusses the obtained results and presents future work.
We assume that the reader is familiar with Colored Petri net [14].

II. Motivations

The Petri nets can be considered as graphic and mathematical tools of modeling and analyzing the discreet system, particularly the competitive, parallel and non-determinist ones. In the field of MAS, the previous works of the Petri nets concentrated on their uses and not on the creation of the new tools and platforms. The goal of our work is to develop a platform which generates automatically models for multi-agent system using CPN. The system in question must be described in an intermediate language. We find in literature two classes of specification languages [15][16]. The first allows the definition of agent and its behavior (e.g., *AgentSpeak* [17]) and the other describes the system environment (e.g., ELMS [18]). Therefore, the definition of a new language including both aspects is necessary. We propose, then, a new language based on XML. The use of XML has many advantages:

- **Universality**: The adoption of a simple and powerful syntax which allows the representation of the most generic models with hierarchical elements, attributes and textual content.
- **Interoperability**: Thanks to their universal syntax, XML documents are easily transportable and readable between systems.
- **Independence between models and data**: We can write an XML document without resorting ever to a schema. If we need to validate the document, we can build a schema afterward.

III. Multi-agent systems (MAS)

According to Weiss [19], agents are computational systems situated in some environment, and are capable of autonomous action in this environment in order to meet their design objectives. Agents perceive and interact with each other via the environment, and they act upon it, so that it reaches a certain state where their goals are achieved. Consequently, the MAS environment consists of a set of states $S = \{s_1, s_2, ...\}$, where an agent can undertake a set of actions $A = \{a_1, a_2, ...\}$ and perceive a set of percepts $P = \{p_1, p_2, ...\}$. Therefore, environment modelling is an important issue in the development of multi-agent systems. Although, some multi-agent systems may be situated in an existing environment, in agent-based simulations, the environment is necessarily a computational process too, so modelling multi-agent environments is always an important issue. For this objective, we present in the next section Multi-Agent System Description Language, a language used for the specification of multi-agent environments.

IV. Multi-Agent System Description Language

In this section, we introduce the main aspects of the language we defined for the specification of the multi-agent system and its environment. The language is called Multi-Agent System Description language (MASDL). MASDL is inspired from the Environment Description Language for Multi Agent Simulation (ELMS) language [18], which is an XML-based language that provides the ability to describe multi-agents.
The syntax and the various components of our language are given below. The validation of the syntax is done using World Wide Web Consortium (W3C) scheme which is a grammar defined in XML formalism.

- **MAS general structure:** MAS specification contains the name of the system, a list of agents, a set of objects (system environment), a list of states (agents states and objects states) and finally a list of actions may be performed by agents. The code sample Listing 1 gives the general structure of the system.

Listing 1: **MAS description structure**

```
<MAS NAME = "">
        <AGENTS_LIST>
                <AGENT NAME = "">
                </AGENT>
        </AGENTS_LIST>
        <OBJECTS_LIST>
                <OBJECT NAME = "">
                <OBJECT></OBJECTS_LIST>
        <STATES_LIST>
          <AGENT_STATE_LIST></AGENT_STATE_LIST>
          <OBJECT_STATE_LIST></OBJECT_STATE_LIST>
        </STATES_LIST>
        <ACTIONS_LIST>
                <ACTION NAME = ""></ACTION>
        </ACTIONS_LIST>
</MAS>
```

- **Agent description:** This description contains the name of the agent, a list of its attributes (agent properties), the current state and list of actions. The following example in Listing 2, defines an agent named *agent1* which has an attribute *prop1* of type *type1* with a value *val1*. The *agent1* has *state_agent1* like initial state and can perform *action1* and *action2*.

Listing 2: **Agent description example**

```
<AGENT NAME = "agent1">
    <ATTRIBUTES>
          <ATT NAME= "prop1"
               TYPE="type1"
               VALUE = "val"/>
    </ATTRIBUTES>
    <CURRENTSTATE>
          <ITEM NAME = "state_agent1"/>
    </CURRENTSTATE>
    <ACTIONS>
          <ITEM NAME = "action1"/>
           <ITEM NAME = "action2"/>
    </ACTIONS>
</AGENT>
```

- **Resources description** This concept allows the specification of the different objects in the MAS environment (all the entities in the environment that are not agent). An object class includes its name, the current state of the object, its identifier and the available quantity (a negative amount is used in case where the amount is unlimited).

Listing 3: **Example of object description**

```
<OBJECTS_LIST>
     <OBJECT NAME = "Objet1">
          <ATTRIBUTES>
                <ATT NAME= "quantity"
                         TYPE= "int"
                         VALUE="100"/>
          </ATTRIBUTES>
          <CURRENTSTATE>
                <ITEM NAME = "state_res"/>
          </CURRENTSTATE>
     </OBJECT>
</OBJECTS_LIST>
```

The code sample Listing 3 above, defines a resource called *object1*. This object has an integer attribute representing the number of units available in the system and current state *state_res*.

- **State description** The state is defined in the tag *<AGENT_STATE_LIST>* when it relates to agents and in the tag *<OBJECT_STATE_LIST>* when it concerns resources. A state is described by its name and an informal description given by the designer, it corresponds to the semantics of the state. A state represents a situation in which an agent or a resource can be, during the running of the system. In Listing 4, the code exemplifies definition of two states (one for agent state and the second for resource one).

Listing 4: **Description of states**

```
<STATES_LIST>
        <AGENT_STATE_LIST>
                <STATE NAME = "state_agent1">
                <DESCRIPTION></DESCRIPTION>
                 </STATE>
         </AGENT_STATE_LIST>
        <OBJECT_STATE_LIST>
                <STATE NAME = "state_res">
                <DESCRIPTION></DESCRIPTION>
                 </STATE>
         </OBJECT_STATE_LIST>
 </STATES_LIST>
```

- **Action description** The description of the action includes its name, its content and an informal description. *Content* specifies the agents that are involved in the execution of the action and also the potential resources (objects) needed. The agents specification includes their name, input and output states. An action can be executed by an agent, if it is in the defined input state. These states represent the preconditions of the action.
Resources can also be instantiated or removed by action. The specification of resources including their type, input and output states and the number of units to subtract (*sub_quantity_entry*) or to add (*add_quantity_exit*).

Listing 5: **Action description**

```
<ACTIONS_LIST>
        <ACTION NAME = "action1">
         <CONTENT>
                <ACTIONS_AGENT>
                        <ACTION_ITEM NAME= "Ag"
                        ENTRYSTATE ="state1_Ag"
                        EXITSTATE="state2_Ag"/>
                 </ACTIONS_AGENT >
                <ACTIONS_OBJECT>
                        <ACTION_ITEM NAME= "Res"
                        ENTRYSTATE="state1_Res"
                        EXITSTATE="state2_Res"
                        SUB_QUANTITY_ENTRY= "2"
                        ADD_QUANTITY_EXIT="5"/>
                 </ACTIONS_OBJECT>
         </CONTENT>
        <DESCRIPTION>
                <!-- Informel Description -->
         </DESCRIPTION>
   </ACTION>
</ACTIONS_LIST>
```

In the example above Listing 5, an action named *action1* is defined and has as a precondition: the agent *Ag* must be in the state *state1_Ag* and the object *Res* in the state *state1_Res*. As a result of the execution of this action, the agent *Ag* will be in the output state *state2_Ag* and the resource *Res* in *state2_Res* state with the production of three units of this resource.

V. Automatic Multi-Agent Modeling Using CPN

The objective of this section is to give an algorithm allowing to transform a description of multi-agent system to Colored Petri net models. The CPN models obtained are written in a XML based language with a specific syntax which we call *Petri Net Description Language* (PNDL).

A. *Transformation algorithm*

The transformation algorithm 1 allows to generate automatically CPN models of the described system. The important steps of the algorithm are given in Fig. 1. Based on MASDL language, the system is defined by a set of states $S = \{s_1, s_2, ...\}$ and agent is able to perform a set of actions $A = \{a_1, a_2, ...\}$. The execution of an action causes changes in the environment.

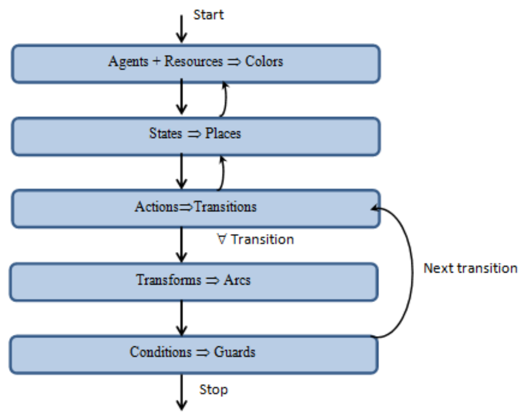


Fig. 1: Schema of the modeling process.

- **Algorithm assumptions:** We assume that our algorithm has as input and output data the following sets, described in the Fig. 2, which are calculated from the MASDL system specification.

Name	Description
AG	Set of agent
RE	Set of resources
SA	Set of agents states
SR	Set of resources states
AC	Set of actions
P	Set of places
T	Set of transitions
Arc	Set of Arcs
C_1	Color with the structure $< id, state >$
C_2	Color with the structure $< id, state, quantity >$

Fig. 2: Algorithm's input and output data

The algorithm also uses a set of predefined functions, the definition of which is as follows:

- ◦ *Act* : $AG \to AC$: *Act(a)* allows to give all the actions which can be made by the agent a,
- ◦ $Act_R : RE \to AC$: $Act_R(r)$ calculates all actions that affect the resource r,
- ◦ $Entry\_Agent : AG \otimes AC \to SA$: Returns the entry state of one agent to undertake an action
- ◦ $Entry\_object : RE \otimes AC \to SR$: Returns the entry state of one resource to undertake an action states,
- ◦ $Exit\_Agent : AG \otimes AC \to SA$:Returns the exit state of one agent after action execution,
- ◦ $Exit\_object : RE \otimes AC \to SR$: Returns the exit state of one resource after action execution,
- ◦ $Sub\_Object : RE \otimes AC \to N$: Gives the number of units to subtract from one resource after action execution,
- ◦ $Add\_Object : RE \otimes AC \to N$: Gives the number of units to add to one resource after action execution,
- ◦ $Create\_Place()$: Allow to create places,
- ◦ $Create\_Transition()$: Allows to create transitions,
- ◦ $Create\_Arc()$: Creates arcs connecting places to transitions or vice versa.

Algorithm 1 Petri net Generator

```
$P \leftarrow \emptyset$
$T \leftarrow \emptyset$
for each $sa \in SA$ do
    $Create\_Place(p_{sa})$
    $P \leftarrow P \cup p_{sa}$
end for
for each $sr \in SR$ do
    $Create\_Place(p_{sr})$
    $P \leftarrow P \cup p_{sr}$
end for
for each $c \in Act(a)$ do
    $Create\_Transition(t_a)$
    $T \leftarrow T \cup t_a$
end for
for each $a \in AG$ do
    for each $c \in Act(a)$ do
        $sa \leftarrow Entry\_Agent(a,c)$
        $sa' \leftarrow Exit\_Agent(a,c)$
        $Create\_arc(p_{sa},t_a)$ with color function $1/ < a,sa >$
        $Create\_arc(t_a,p_{sa'})$ with color function $1/ < a,sa' >$
    end for
end for
for each $r \in RE$ do
    for each $c \in Act_R(r)$ do
        $sr \leftarrow Entry\_Agent(r,c)$
        $sr' \leftarrow Exit\_Agent(r,c)$
        $Create\_arc(p_{sr},t_c)$ with color function $Sub\_Object(r,c)/ < r,sr,quantity >$
        $Create\_arc(t_a,p_{sa'})$ with color function $Add\_Object(r,c)/ < r,sr',quantity >$
    end for
end for
```

- **Initial marking:** the initial state is calculated as:
  1) If an agent a is initially in the state sa, we put one token of color $< a,sa > \in C_1$, in the place p_{sa};
  2) If an resource r is initially in the state sr, we put one token of color $< r,sr,quantity > \in C_2$, in the place p_{sr};

B. Output format

We propose an XML-based language for the description of the models generated by the algorithm. We entitle our language Petri Net Description Language, which is based on the tags $< PLACES >$, $< TRANSITIONS >$ and $< ARCS >$ to describe the model and on $< COLORS >$ and $< TOKENS >$ to give its marking. The general structure of the language is presented in the following Listing 6:

Listing 6: Description of CPN Model

```
<RDPC NAME = ”RdP_Example”>
        <PLACES>
                <PLACENAME = ”p1”/>
        </PLACES>
        <TRANSITIONS>
                <TRANSITIONNAME = ”t1”/>
        </TRANSITIONS>
                <ARCS>
                <PRE_ARCS>
                        <ARC FROM ”p1” TO ”t1”>
                        <WEIGHT COLOR = ”c1” PRE = ”1”/>
                        </ARC>
                </PRE_ARCS>
                <POST_ARCS>
                        <ARC FROM ”t1” TO ”p1”>
                        <WEIGHT COLOR = ”c1” POST = ”1”/>
                        </ARC>
                </POST_ARCS>
        </ARCS>
        <COLORS>
                <COLOR NAME = ”c1”>
                        <ITEM NAME = ”id” VALUE = ”01”/>
                </COLOR>
        </COLORS>
        <TOKENS>
                <TOKEN COLOR = ”c1” PLACE = ”p1”/>
        </TOKENS>
</RDPC>
```

VI. Running MASDL environment

The use of XML provides various advantages, wide range of XML tools are currently available and it can be useful for the future development. The validation of the description is done using W3C scheme. For the implementation of our tool, we chose Java, which allows us to use Java Architecture for XML Binding (JAXB) and Application Programming Interface (API) to create XML application data. The global architecture of our application is shown in Fig. 3.

Our tool allows users to introduce environment specifications from a graphical interface, as shown in Fig. 4. With this interface, users do not need to deal with the language syntax but just fill the different fields.

Filled fields will be checked and compiled to generate the corresponding XML file, as shown in Fig. 5, on which the transformation algorithm will be applied. To generate a

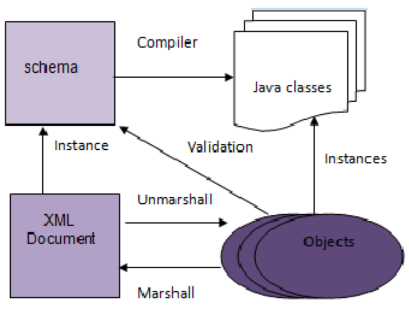
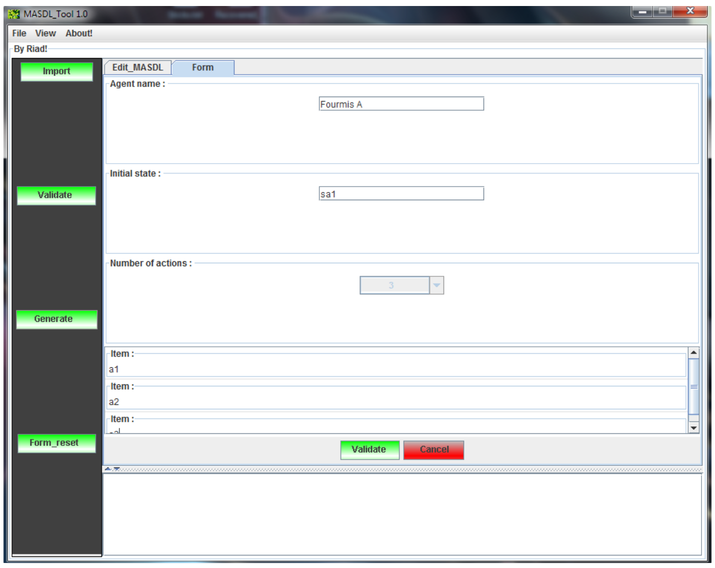


Fig. 3: The global architecture

graphical representation of the Petri net model, as shown in Fig. 6, we use the GraphViz tool [20].



Agent name : Fourmis A

Initial state : sa1

Number of actions : 3

Item : a1

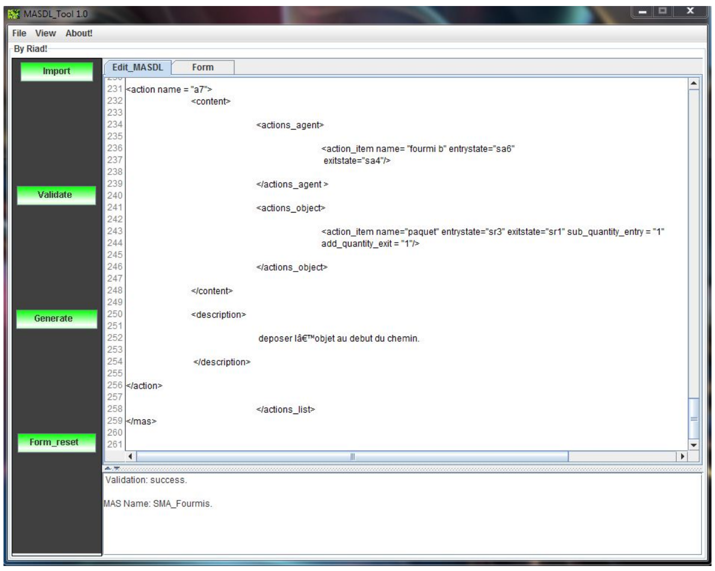
Item : a2

Item :

Fig. 4: Tool interface

VII. CONCLUSION AND FUTURE WORKS

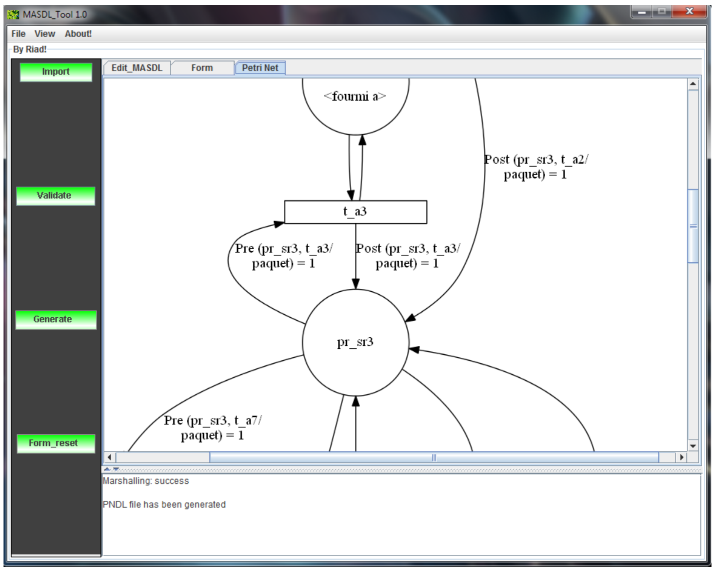
In this paper, we introduced the MASDL language, used for the specification of the agents and their environment. The language is based on XML and is independent of the agent runtime platform and implementation language. We defined also transformation rules to obtain formal models from the system specification to analyze and verify the described multi-agent system. We plan in our further work to connect our tool to another verification tool, as CPN Tool or GreatSPN for the general properties verification (deadlock, boundedness, etc.). We will focus mainly on extending our model by introducing the temporal dimension in order to perform a quantitative analysis and compute MAS system performances (average waiting time, average resources available, etc.).



```
<action name = "a7">
              <content>

                            <actions_agent>

                                          <action_item name= "fourmi b" entrystate="sa6"
                                           exitstate="sa4"/>

                            </actions_agent >

                            <actions_object>

                                          <action_item name="paquet" entrystate="sr3" exitstate="sr1" sub_quantity_entry = "1"
                                          add_quantity_exit = "1"/>

                            </actions_object>

              </content>

              <description>

                            deposer lâ€™objet au debut du chemin.

               </description>

</action>

                            </actions_list>
</mas>
```

```
Validation: success.

MAS Name: SMA_Fourmis.
```

Fig. 5: MASDL file exemple


MASDL_Tool 1.0
File View About!
By Riad!
Import
Validate
Generate
Form_reset
Edit_MASDL
Form
Petri Net
<fourmi a>
Post (pr_sr3, t_a2/ paquet) = 1
t_a3
Pre (pr_sr3, t_a3/ paquet) = 1
Post (pr_sr3, t_a3/ paquet) = 1
pr_sr3
Pre (pr_sr3, t_a7/ paquet) = 1

```
Marshalling: success

PNDL file has been generated
```

Fig. 6: Petri net generation

REFERENCES

[1] M. Greaves, V. Stavridou-Coleman, and R. Laddaga, “Guest editors' introduction: Dependable agent systems,” *IEEE Intelligent Systems, NJ, USA*, vol. 19, September 2004, pp. 20-23.

[2] S. S. Heragu, R. J. Graves, B.-I. Kim, and A. St Onge, “Intelligent agent based framework for manufacturing systems control,” *Trans. Sys. Man Cyber. Part A, NJ, USA*, vol. 32, no. 5, September 2002, pp. 560-573.

[3] J. Ferber, *Les systèmes multi-agents vers une intelligence collective*. Inter-Editions, 1995.

[4] H. Mouratidis, M. Kolp, P. Giorgini, and S. Faulkner, “An architectural description language for secure multi-agent systems,” *Web Intelli. and*

Agent Sys., Amsterdam, The Netherlands, vol. 8, no. 1, January 2010, pp. 99-122.

[5] M. Dziubiński, “Complexity of multiagent bdi logics with restricted modal context,” in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, ser. AAMAS ’11. International Foundation for Autonomous Agents and Multiagent Systems, Taipei, Taiwan, 2011, pp. 1171–1172.

[6] S. Maalal and M. Addou, “A new approach of designing multi-agent systems,” *CoRR*, vol. abs/1204.1581, 2012.

[7] L. J. B. Ayed and F. Siala, “Event-b based verification of interaction properties in multi-agent systems,” *JSW*, vol. 4, no. 4, 2009, pp. 357-364.

[8] J. R. Celaya, A. A. Desrochers, and R. J. Graves, “Modeling and analysis of multi-agent systems using petri nets,” *JCP*, 2009, pp. 981-996.

[9] C. Balague, “Multi-agent system in marketink: Modelisation by petri net,” Ph.D. dissertation, École des Hautes études Commerciales, 2005.

[10] M. K. Gazdare, “Heuristic optimization of distributed problem storage containers in port,” Ph.D. dissertation, ECOLE CENTRALE DE LILLE, 2008.

[11] A. El Fallah-Seghrouchni, S. Haddad, and H. Mazouzi, “Protocol engineering for multi-agent interaction,” in *International Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW), Valencia, Spain*, 1999.

[12] D. Boukredera, S. Aknine, and R. Maamri, “Modeling temporal aspects of contract net protocol using timed colored petri nets,” in *STAIRS*, December 2012, pp. 83–94.

[13] S. Khosravifar, “Modeling multi agent communication activities with petri nets,” *International Journal of Information and Education Technology, Singapore*, vol. 3, no. 3, September 2013, pp. 310–3014.

[14] K. Jensen, *Coloured Petri nets: basic concepts, analysis methods and practical use, vol. 2*. London, UK, UK: Springer-Verlag, 1995.

[15] M. M. Dastani, C. M. Jonker, and J. Treur, “A requirement specification language for configuration dynamics of multi-agent systems,” *International Journal of Intelligent Systems.*, vol. 19, 2004, pp. 277–300.

[16] M.-P. Huget, “Agent uml notation for multiagent system design,” *IEEE Internet Computing, Piscataway, NJ, USA*, vol. 8, no. 4, 2004, pp. 63–71.

[17] A. S. Rao, “Agentspeak(l): Bdi agents speak out in a logical computable language,” in *Proceedings of the 7th European workshop on Modelling autonomous agents in a multi-agent world : agents breaking away: agents breaking away*, ser. MAAMAW ’96. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1996, pp. 42–55.

[18] F. Y. Okuyama, R. H. Bordini, and A. C. da Rocha Costa, “Elms: an environment description language for multi-agent simulation,” in *Proceedings of the First international conference on Environments for Multi-Agent Systems*, ser. E4MAS’04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 91–108.

[19] G. Weiss, Ed., *Multiagent systems: a modern approach to distributed artificial intelligence*. Cambridge, MA, USA: MIT Press, 1999.

[20] Graphviz - graph visualization software. http://www..graphviz.org. Retrieved: June, 2013.

Data From Configuration Management Tools As Sources For Software Process Mining

Jana Šamalíková, Rob Kusters, Jos Trienekens, Ton Weijters,
IS, IE&IS
University of Technology Eindhoven
Eindhoven, The Netherlands
j.j.m.trienekens@tue.nl

Abstract—**Process mining has proven to be a valuable approach that provides new and objective insights into processes within organizations. Based on sets of well-structured data, the underlying 'actual' processes can be extracted and process models can be constructed automatically, i.e., the process model can be 'mined'. Successful process mining depends on the availability of well-structured and suitable data. This paper investigates the potential of software configuration management (SCM) and SCM- tools for software process mining. In a validation section, data collected by a SCM tool in practice are used to apply process-mining techniques on a particular software process, i.e., a Change Control Board (CCB) process in a large industrial company. Application of process mining techniques revealed that although people tend to believe that formally specified and well-documented processes are followed, the 'actual' process in practice is different. Control-flow discovery revealed that in the CCB process in most of the cases, i.e., 70%, an important CCB task 'Analysis' was skipped.**

Keywords-software configuration management; process mining; validation

I. Introduction

Software process improvement is a cyclic activity during which improvements are planned, applied and their impact is analyzed. Before planning improvements, the current state of the software process has to be assessed. Nowadays, process assessment is based on process descriptions obtained from quality manuals and process standards, as well as on information that is derived from interviews and brainstorm sessions with representative software developers [1]. A promising alternative way to obtain close-to-the-reality process descriptions is process mining. Process mining has proven to be a valuable approach that provides new and objective insights into the way processes are actually carried out within organizations [2], [3], [4]. In a number of case studies, event logs were created from data that were automatically recorded by process-enactment systems. Based on sets of well-prepared data, the underlying 'actual' processes can be extracted and process models can be constructed automatically, i.e., the process model can be 'mined'. Software developing organizations use various types of software tools to support and manage their software development processes, e.g., configuration management tools, inspection tools and testing tools. In this paper, we will focus on software configuration management (SCM) and SCM tools as a good example of process support tools. "Software Configuration Management (SCM) tools are 'the' real process-centered tools due to their ability to model, support and enact the processes by which all software developers are supposed to manipulate the product" [6]. In SCM tools, several types of data about the development processes are stored, such as data about the tasks or activities that are carried out by the developers, and data about the creation of and the changes on software components. Various SCM tools can provide so-called 'audit trails' of the collected data. However, data logged during software development are often not intended for process mining. It is necessary to investigate whether the data logged by SCM tools could in principle be used for process mining. For example regarding process mining, these data should have a notion of a process (e.g., time-stamp data), but they should also offer the possibility to identify particular objects with particular attributes that are handled by a process. In process mining, these objects are called 'cases'. The aim of this paper is to evaluate selected SCM tools regarding their potential to provide data for process mining. The paper is organized as follows. In Section 2, a structured overview is derived of the data types that are required for the application of process mining techniques. Section 3 identifies, from the viewpoint of process mining, the data that play a central role in the software processes, which are supported by SCM. Subsequently, Section 4 investigates the potential of selected SCM tools to provide the required data for software process mining. Section 5 finalizes the paper with conclusions and points to future work to be done.

II. Process mining techniques and the data that they require

Process mining techniques attempt to extract non-trivial and useful process information from so-called audit trails. In order to be useful for process mining an audit-trail has to be transformed into an event log. Currently a variety of process mining techniques is available and has been applied in practice [7]. These mining perspectives and mining techniques are described in more detail in the next subsections.

A. The control-flow perspective and the data required

From the control-flow perspective process model discovery techniques are applied to discover a process model that specifies the relations between the activities in an event log. The resulting mined process model is a 'real' *model* that depicts the possible flows that were followed by particular cases in an event log. Subsequently, conformance checking can be carried out in order to compare the 'official' process model with the 'real' process model as it is stored in the event log. The data requirements for process model discovery and conformance checking are: an event log containing ordered sequences of events (i.e., of type 'start' or 'end') where each event refers to a case and each event refers to an activity.

B. The performance perspective and the data required

The performance perspective focuses on the performance of processes. Mining from the performance perspective calculates the timeliness of cases, the execution times of tasks, and reveals the bottlenecks in processes by calculating waiting times (synchronization time). The throughput time calculation technique can be applied if a timestamp of an end event is recorded. This mining technique implements an event log replay and therefore a 'real' process model is required that captures the behavior in the event log. Applying performance sequence analysis techniques reveals sequence patterns that are present in an event log. Based on this application, it is possible to determine which sequence patterns are common and which patterns are less frequent. The data requirements are respectively: an event log containing ordered sequences of events where each event refers to a case and each event refers to an activity, and timestamps of the start and end event of activities.

C. The organizational perspective and the data required

The organizational perspective in process mining focuses on the analysis of the interrelations among persons (or groups of individuals) who are performing the activities in a process, i.e., their social network. Social network analysis techniques focus on the discovery and examination of the social interrelations. Several types of social network mining are possible with different types of results; such as a work transfer model, a subcontracting model, a collaboration model, and a similar activity model. Data requirements are respectively: an event log containing ordered sequences of activities, where each activity refers to a case, and each case refers to an activity and to its originator.

D. The case perspective and the data required

A case is an 'object' that is being handled by a process. Case attributes represent various case properties and together they specify a case. Two types of case attributes exist: the stable attributes (e.g., a defect id. or a phase in which a defect was detected) and the dynamic attributes, which become available during a process (e.g., a defect priority). Both types of case attributes enable process mining from the case perspective. The case perspective shows the process based on the case types (i.e., a set of cases with the same or similar attributes), discovering the data dependencies that affect the routing of a case. Within the case perspective decision analysis techniques are being applied. These techniques focus at the way case attributes influence the choices that are being made in the process, based on past process executions. The data requirements of decision analysis techniques are respectively: an event log containing ordered sequences of events where each event refers to a case and each event refers to an activity, and the case attributes that are modified during the execution of activities. Table I summarizes the event log data requirements per process mining technique.

TABLE I. EVENT LOG DATA REQUIREMENTS

Required data / Process mining techniques	Event log data					
	Case id	Activity	Event type	Timestamps	Originator	Additional attributes
Process model discovery	x	x	x			
Conformance checking	x	x	x			
Throughput time calculation, bottleneck analysis	x	x	x	x		
Performance sequence analysis	x	x	x	x		
Social network analysis	x	x	x		x	
Decision analysis	x	x	x			x

III. SOFTWARE PROCESSES (AND THEIR CASES) THAT ARE SUPPORTED BY SOFTWARE CONFIGURATION MANAGEMENT

A key aspect of SCM in software development is version management. A software component put under version control is called a configuration item (CI). Besides managing the versions of CIs in the software development process, SCM supports the change control process, the problem management, and the requirements management process. We selected these four processes as well-structured and formally described processes. In the following, we describe these processes in more detail and we will investigate whether the recorded data, i.e., their primary 'cases', can be used for process mining.

E. *Software development process*

The software development process contains the activities of developers that are performed during the software lifecycle. The process contains activities for requirements analysis, design, coding, integration, testing, and installation of software products. During these activities, various software components or CI's are produced [5]. Components are put under version control and are controlled by SCM. Therefore, we identify a software component as a case that is handled by the software development process.

F. *Change control process*

The change control process manages the change requests to a software product. Configuration management is in general under control of a CCB [1]. A CCB coordinates changes made to CI’s. The CCB tracks and records the status of each change request, e.g., labeled as defects, from its entry until its exit from the CCB process. The CCB distributes tasks related to the required changes of CI’s and evaluates the outcomes of the executed tasks with respect to the requests. A so-called audit trail is available, where each modification, the reason for the modification, and the authorization of the modification can be traced. Based on the foregoing we identify a change request as a case that is handled by the change control process.

G. *Problem resolution process*

The problem resolution process is established for handling problems detected in software products and activities. The process ensures that all detected problems are promptly reported and entered into the problem resolution process. A unique identification of a problem is assigned during this activity. A problem report is to be used as part of a close loop process, from detection of a problem through investigation, analysis, and resolution, and later for trend detection across problems. Status is tracked and reported, and records of problem reports are maintained. A problem report can be considered as a case that is handled by the problem resolution process.

H. *Requirements management process*

Requirements management is the process of eliciting, documenting, analyzing, tracing, prioritizing and agreeing on requirements with the relevant stakeholders. Requirements management includes the ensuring that requirements are well defined, agreed to by relevant parties and modified in accordance with defined procedures. The modification procedures must ensure that later changes are incorporated properly and that project plans are updated accordingly. In SCM, requirement specifications are considered as common CI's.

TABLE II. CASE IDENTIFICATION

Processes supported by SCM	Cases handled by the processes
Software development process	Component
Change control process	Change request
Problem resolution process	Problem report
Requirements management process	Requirement specification

Therefore, we consider a requirement specification as a case that is handled by the requirements management process. Table II summarizes the software development processes that are supported by SCM together with the cases that are being handled by these processes. In the next section, we will investigate the potential of four selected SCM tools with respect to the data that they can provide to software process mining.

IV. SCM TOOLS AS DATA SOURCES FOR PROCESS MINING

Nowadays, a variety of SCM tools exists. We can distinguish SCM tools that offer particular basic functionalities, such as version management, but also ‘more rich’ SCM tools that are called ‘SCM suites’. The latter support a range of additional functionalities such as change request control, problem resolution control, build and release management, requirements management, project and task management, and even workflow management. SCM tools assist developers in their collaborative work, support the maintenance of software products, storing their history, providing a stable development environment and coordinating simultaneous product changes [8]. Data recording in SCM tools is reflected by so-called audit trails. We will investigate the content and the data structure of these audit trails, and to what extend these data can be used as event log data for software process mining. We will analyze selected SCM tools with respect to their ability to provide data for process mining techniques.

I. *SCM tool selection and analysis*

Based on a survey we selected the two most commonly used SCM tools in order to analyze their audit trails with respect to process mining. These tools, respectively Subversion and Microsoft Visual SourceSafe, are examples of basic version management tools. Additionally, we selected CM Synergy & Change Synergy and HP Quality Center tools as examples of the more ‘richer’ tools. These tools provide also support for change request control, problem resolution control and requirements management. Regarding the first mentioned SCM suite we identified the part that is called Rational Synergy as a basic version management tool. Consequently, we had three basis version management systems. In our analysis, we will focus on the one hand on particular software processes that are supported by these tools, respectively: the software development process, the change control process, the problem resolution process and the requirements management process (see section III, Table II). On the other hand, we will focus on particular data that are recorded by the selected SCM tools, respectively: case id., task, event type, time stamp, task originator, additional attributes of cases, as identified in Section II, Table I.

1) *Basic version management tools*

Basic version management tools offer basic functions of version management such as managing the evolution of configuration items, file sharing, controlling concurrent work, history tracking, and security and access control. We analyzed three basic version management tools more in depth, respectively Subversion, Microsoft Visual SourceSafe, and Rational Synergy (the basic version management part of the 'SCM suite' CM Synergy and Change Synergy). Because of similarities in the results we will describe in this paper only the analysis results of Subversion. Subversion is a basic open-source file-based version management tool, which tracks the changes to files and directories under version control. After a revision, i.e., a change of a file is made, the file is committed to a repository of files, and a log-entry is created in the tool log (the so-called 'change log'). Subversion supports the software development process. The software development process is reflected by sequences of document changes, with respect to particular components, in a change log. Table III shows the identified event log elements. Please note that there are no additional attributes of cases recorded.

TABLE III. EVENT LOG ELEMENTS IN THE SUBVERSION LOG

Event log element	Case id	Acti vity	Event type	Time stamp	Origi nator
software develop-ment process	com-ponent	docu-ment type	end	of check-in	user check-ing-in

In the following subsection, we will analyze two 'SCM suites', respectively CM Rational Synergy and HP Quality Center.

2) Integrated SCM tools

The so-called 'SCM suites' are in fact integrated tools that incorporate various functions of version management in combination with enhanced functionalities such as change control, build management, problem issue management, process and workflow management, baseline management and requirements management.

a) CM Rational Synergy and Rational Change

This is a SCM tool that provides integration and synergy between the different SCM functions. It is an integrated tool that supports distributed development on a unified change, configuration and release management platform. The 'suite' consists of a Rational Synergy part and a Rational Change part. The Rational Change part is a web-based, integrated, change management tool for tracking and reporting changes of cases. Rational Change supports in particular three processes, respectively the change control process, the problem resolution process and the requirements management process. The cases handled by these processes are respectively change requests, problem reports and requirements specification requests. The identified event log elements are shown in Table IV. Case id corresponds to a case, Activity corresponds to a status adjustment of a case, Event type: 'start' and 'end', Timestamp corresponds to a commit date and time, additional attributes of cases such as priority, severity. Regarding originator: an actual originator of an activity is not provided.

TABLE IV. EVENT LOG ELEMENTS IN THE RATIONAL CHANGE LOG

Event log element	Case id	A cti vit y	Eve nt Typ e	Time stamp	Add. attributes of cases
change control process problem resolu-tion process req. mngt. process	change control request pro-blem report req. spec. request	s t a t u s	start end	of activities e.g.,: submit analysis resolu-tion evalua-tion	e.g.,: priority, severity, phase detected, phase caused

b) HP Quality Center

HP Quality Center supports the change control process, the problem resolution process and the requirements management process. As shown in Table V, the event-log elements are identified as follows: for each of the cases a Case identification number is stored, Activity corresponds to the status adjustment of these cases (the status of a change request is adjusted as a result of executing an activity), Event-type: start or end, Timestamps: timestamp of the end event of a related activity. Additional attributes: for each supported process a rich variety of additional case attributes are recorded. These attributes describe the cases in more detail than the previously discussed Rational Change tool and as a consequence allow for more detailed analysis. Pleae note that although the originator is not shown in Table V, information on person/department executing the activities is recorded.

J. *SCM tools and the process mining techniques that can be supported by them*

Data needed for process mining are (partially) available in the so-called audit trails of the selected and analyzed SCM tools. These process data can be used as a basis for the construction of the event logs needed for process mining. We discovered that all the selected SCM tools, provide the minimal data requirements for process mining, i.e., the identification of a case, an activity, and the type of an event.

TABLE V. EVENT LOG ELEMENTS IN THE HP QUALITY CENTER LOG

Event log element	Case id	Activity	Event type	Time stamp	Add. Attributes of cases
Change Control Process	change request	status	start, end	history date, end event of a related activity	change category, root cause, priority, phase detected, estimated cost, responsible team, affected component
Prob. Resolution Process	problem report		end		problem category, related component, criticality, phase detected, responsible team
Req. mngt. Process	spec. request		end		priority, linked test, responsible team, related component

Furthermore, both the basic version management tools and the 'SCM suite' HP Quality Center provide originator information. The 'SCM suites' also provide various additional case attributes. Based on our findings in the foregoing sections, we summarize our SCM tool analysis results in Table VI, i.e., which shows the interrelations between the SCM tools (and the supported processes) and the process mining techniques. The mining techniques process model discovery, and conformance checking can be supported by each of the selected SCM tools. The process mining technique decision analysis is only possible with the SCM tools that provide additional case attributes, i.e., Rational Change and HP Quality Center. Rational Change is the only SCM tool that records both Start and End events of tasks, thereby enabling the process mining techniques Performance Sequence Analysis, Throughput Time Calculation and Bottleneck Analysis. The process mining technique social network analysis is possible with each of the SCM tools that record task originator (i.e., Subversion, VSS, Rational Synergy, and HP Quality Center).

TABLE VI. INTERRELATIONS BETWEEN PM TECHNIQUES AND SCM TOOLS

Process mining technique / SCM tool (and process supported)		Process model discovery	Conformance checking	Decision analysis	Throughput time calculation, bottleneck analysis	Performance sequence analysis	Social network analysis
Subversion	software development process	x	x		x		x
Visual SourceSafe	software development process	x	x				x
Rational Synergy	software development process	x	x				x
Rational Change - Change Synergy	change control, problem resolution, requirements management process	x	x	x	x	x	
HP Quality Center	change control, problem resolution, requirements management process	x	x	x			x

Summarizing we can state that the selected SCM tools meet important event log requirements for process mining. The basic version management tools, such as Subversion, provide software development process data. However, the integrated 'SCM suites', such as Rational Synergy, support particular software processes as well, respectively the change control process, the problem resolution and the requirements management process.

V. VALIDATION

In order to validate our approach, we used the data collected during ten middleware embedded-software projects of a large industrial company in The Netherlands. The detailed results of the validation are presented in [1]. In this section we address some main results of the case study. The industrial company develops software components for consumer electronic devices. The process under study is the CCB process. The CCB is an organizational unit that handles problem reports, change and implementation requests identified during software development. These are further referred to as defects. A CI's defect is detected and submitted. The developer assigns attributes to the defect (e.g., priority, severity). After that, the defect is either processed by the main sequence of tasks Analysis, Resolution, Evaluation, Conclusion or it is evaluated by the CCB. If the CCB evaluates the defect, it sends the defect to a required task depending on the need, with the following possibilities: the defect is redirected to the Concluded task in case the defect is found duplicated, expected to be repaired in a next release, or out of the scope of the functionality required. The defect is redirected to tasks Analysis, Resolution or Evaluation depending on the need. When the task Analysis, Resolution or Evaluation is completed, one of the four possibilities is chosen: if the task's execution is successful, then an important defect is directed to the CCB and it waits to be redirected again to the next task; if the task's execution is successful, then a less important defect continues with the next task of the main sequence of tasks; if the task was not successfully executed, then an important defect is returned to the CCB for a re-evaluation. Once all the tasks of the CCB process have been successfully carried out, the defect is closed.

The software development team collects defect data and these are stored in the status database recorded by a Rational Change SCM tool. A quality assurance specialist creates copies (snapshots) of the status CCB database content on a weekly basis. The snapshots follow the evolution of the handling of the defects by the CCB. The snapshots include the following information: the date when the snapshot was taken, identification of a subsystem from which the snapshot was taken, identification of the defect, priority and severity of the defect, type of the defect, the actual status of the defect, identification of a team responsible for resolving the defect, timestamps of the start event of the tasks of the main sequence, timestamps of the end event of the tasks of the main sequence and a timestamp of the date of an update of the status database. We studied whether it is possible to use such data to discover and construct underlying process models. We identified the following event log elements. Case id (a defect is identified as a case), a Case id is retrieved from a Problem_nr data field that uniquely identified the defect. Activities are executed when they handle a case during a process. As a result of executing an activity, the status of the defect changes, therefore activities have been derived from the status field. Event-type: in the snapshots, only the event types start and end are used. Timestamp have been extracted from the fields in the snapshots that store time information. Originator is not available due to the fact that the snapshots only provide the information about the team responsible for resolving the defect. Additional attributes are derived from the priority, severity, request type, life-cycle phase during which a defect had been discovered. Based on the identified event log elements, we identified process mining techniques that are possible to be applied to the data. We applied respectively control-flow discovery and conformance checking. We applied the mentioned techniques to the event-log filtered on additional case attributes, namely priority, severity, request type and life-cycle phase. The control-flow discovery revealed that the official CCB process as described above was not followed. Furthermore, conformance checking of the CCB process revealed that in most of the cases (70%) the Analysis task is skipped and the cases are being directly resolved. Moreover, we compared the duration of tasks and the total throughput time during different lifecycle phases. The results showed that the duration of the validation tasks involving external stakeholders are longer than the verification tasks performed without the external involvement. Application of process mining techniques revealed that although people tend to believe that specified and well-documented processes are followed, the real practice is different., and that process mining techniques can provide useful insights into the software development process.

VI. CONCLUSIONS

This paper investigates the application of process-mining techniques in the software development domain. We addressed the suitability of particular software development support tools, i.e., SCM tools, to provide process data. Our research provides an original view at process mining literature from a data requirements perspective: 'different process mining techniques require different data', and can form a basis for the development of new functionalities, i.e., process mining support, to develop a future generation of SCM tools. We addressed some main results of a case study, to indicate which process mining techniques can make use of particular SCM tool data to get in-depth insights into 'actual' software development processes.

REFERENCES

[1] J. Samalikova, R. Kusters, J. Trienekens, T. Weijters, and P. Siemons, "Towards objective software process information: experiences from a case study," Software Quality Journal, vol. 19, Jan. 2011, pp. 101-120.

[2] A. Weijters, W. van der Aalst and A.K. Alves de Medeiros, "Process Mining with the Heuristics Miner Algorithm," BETA Working Paper Series, WP 166, Eindhoven University of Technology, Eindhoven, 2006, pp. 1-18.

[3] W. van der Aalst et al, "Business process mining: An industrial application," Information Systems, 32(5), 2007, pp. 713-732.

[4] A. Rozinat., and W. M. P. van der Aalst, "Decision Mining in ProM", in S Dustdar and A Sheth eds., Business Process Management., Lecture Notes in Computer Science, Berlin, Springer, 2006, pp. 420-425.

[5] J. Samalikova, R Kusters, J Trienekens and T Weijters. "Information gathering in software process assessment," In Proceedings of the Information Systems IADIS Conference, Berlin, Germany, 2012, pp. 43-52.

[6] R. Conradi, A. Fuggetta and M. Jaccheri. "Six theses on software process research". Software Process Technology, 1998, pp. 100-104.

[7] B.F. van Dongen et al, "The ProM Framework: A New Era in Process Mining Tool Support", in Applications and Theory of Petri Nets Lecture Notes in Computer Science, Berlin, Springer, 2005, pp. 444-454.

[8] J. Estublier, "Software Configuration Management: a roadmap", in: Proceedings of the Conference on the Future of Software Engineering, Limerick, Ireland, 2000, pp. 279-289.

Refactoring of Simulink Diagrams via Composition of Transformation Steps

Quang Minh Tran, Benjamin Wilmes
Berlin Institute of Technology
Daimler Center for Automotive IT Innovations (DCAITI)
Berlin, Germany
E-Mail: {quang.tranminh,benjamin.wilmes}@dcaiti.com

Christian Dziobek
Daimler AG
Mercedes-Benz Cars Development
Sindelfingen, Germany
E-Mail: christian.dziobek@daimler.com

***Abstract*—Model-based design has been increasingly adopted by the industry, especially the automotive industry, for the development of embedded software. Today, Matlab/Simulink by The MathWorks is widely employed as a modeling tool in which embedded software is modeled as data flow diagrams consisting of blocks and signals. While refactoring has become an established technique for improving the structure of code in textual programming languages, refactoring Simulink diagrams is relatively unexplored. This paper introduces a technique for specifying and implementing refactoring operations for Simulink diagrams by composing elementary and composite transformation steps. How the transformation steps can be leveraged to specify and implement complex refactoring operations is demonstrated based on the two refactoring examples *Replace Goto/From With Explicit Signals* and *Merge Subsystems*. Our prototypical implementation of a refactoring extension for Simulink is also briefly described.**

***Keywords*-Simulink; Refactoring; Transformation**

I. INTRODUCTION

The model-based design (MBD) paradigm has been widely adopted by the automotive industry to develop embedded software, with Matlab/Simulink [1] by The MathWorks being the defacto standard modeling tool. Using Simulink, software functionality is modeled as data flow diagrams by connecting functional blocks via data-carrying signals. Additional concepts of the Simulink modeling language address practical needs, like the readability of large models. For instance, model fragments can be hierarchically grouped into logical units called subsystems and related signals can be grouped into structured bus signals. The adoption of MBD using Simulink leads to models being central artifacts in development. Due to the continuously increasing software complexity and short development cycles, the creation and maintenance of models have become highly intensive and time-consuming activities.

Refactoring is an established restructuring technique which implies changing the structure of a development artifact without changing its observable behavior. Semi-automated or interactive refactoring operations have been integrated into textual programming environments like Eclipse or Visual Studio. However, at present, refactoring is practically non-existent in the Simulink Editor. The missing support for refactoring in Simulink has two potentially severe consequences. First, the model quality may be compromised if quality-improving model changes are not done due to tight development time, even if the modeler is aware of the structural deficits. Second, refactoring a huge Simulink model manually can be very labor-intensive and error-prone.

Thus, in this paper, we present a modular technique for refactoring Simulink diagrams based on the composition of predefined transformation steps. While the focus of this paper is on the underlying refactoring mechanism, we refer to a previous publication of ours for a wider spectrum of useful refactoring operations for Simulink diagrams [2].

The paper is structured as follows. In Section II, we present our meta-model for Simulink models, which serves as the basis for defining transformation steps. Our mechanism for composing transformation steps is described in Section III. How even complex refactoring operations can be specified and realized by utilizing primitive but powerful transformation steps is shown in Section IV. Insight into our prototypical implementation of the concept as an extension of the Simulink Editor is provided in Section V, followed by a summary of related work in Section VI, and our conclusion in Section VII.

II. SIMULINK META-MODEL FOR REFACTORING

The development of a refactoring technique for Simulink diagrams inevitably requires the existence of a meta-model. Unfortunately, to date, no official meta-model for Simulink diagrams has been published. Hence, we defined our own Simulink meta-model which, for the purpose of refactoring, implicitly meets the following criteria:

1) All necessary structural properties of diagrams that are required by refactorings should be captured, including model hierarchy, signal properties and bus structure
2) Support for incomplete diagrams, such as those with unconnected signals, since some refactorings can be triggered at any time during the modeling
3) Layout information must be captured because the execution of a refactoring operation should preserve the layout as much as possible
4) Establish a degree of granularity that enables local structural changes during a refactoring operation without affecting irrelevant model parts

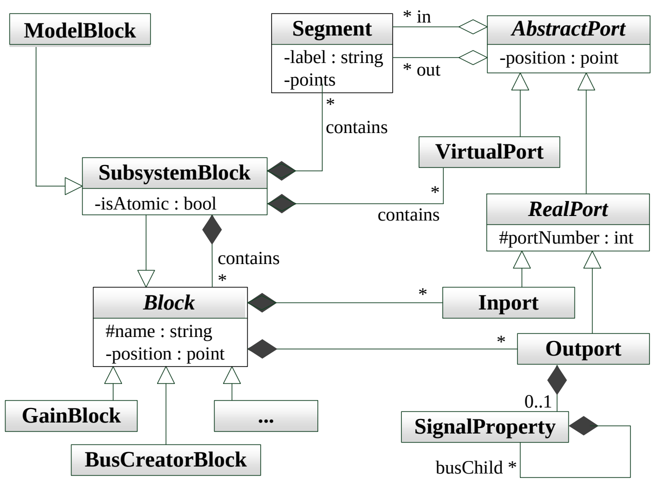


Figure 1. Excerpt of our meta-model for Simulink diagrams

Category	Transformation Steps

Elementary	addBlock(blockType, [pos])
	addInportBlock(destSubsys,[pos])
	addOutportBlock(destSubsys,[pos])
	copyBlock(block, destSubsys, [pos])
	replaceBlock(block, newBlockType)
	deleteBlock(block)
	addSegment(srcPort, targetPort)
	rerouteSegmentToNewTargetPort(seg, newTargetPort)
	rerouteSegmentToNewSourcePort(seg, newSourcePort)
	branchSegmentToNewTargetPort(seg, newTargetPort)

Composite	moveBlocks(blocks, destSubsys, [pos]) deleteBlockWithSignals(block) **addCrossHierarchicalSignal (sourcePort, targetPorts)** rerouteSegmentCrossHierarchicallyToNewTargetPort (sourcePort, newTargetPort) branchSegmentCrossHierarchicallyToNewTargetPort (seg, newTargetPort)

Table I
EXCERPT OF THE TRANSFORMATION STEPS COLLECTION WITH THE STEPS BEING DISCUSSED IN MORE DETAIL MARKED BOLD

Figure 1 shows an excerpt of our meta-model as class diagram. In this meta-model, *Block* is the superclass for all other block types. A *Block* has a unique name in its hierarchical scope and an ordered list of *Inport* and *Outport* instances. Its position is stored in the field *position*. A *Subsystem* is a block that can contain child blocks including other subsystems. An entire model is also a *Subsystem*.

Regardless of whether a signal is completely connected, i.e., constituting an uninterrupted path from one source block to one or more destination blocks, it is divided into one or more segments. A segment connects a source and a target port - which can be of the following types: A *real port* belongs to a block and is either an *inport* (for an incoming signal) or *outport* (for an outgoing signal). A *virtual port* is either a branching point of a signal or an end point of an incompletely connected signal. Both real ports and virtual ports have an (x, y) position. In contrast to a virtual port, a real port has a port number. A segment is called unconnected if its source is a virtual port but not a branching point, or its target is a virtual port.

III. TRANSFORMATION STEPS AND THEIR COMPOSITION

Instead of formulating each refactoring operation individually, we have set the goal to define basic transformation and modification steps that can be aggregated for specifying and implementing complex refactoring operations. As a result, on top of the meta-model in Section II, we have identified a collection of transformation steps (see Table I). A transformation step modifies an instance of the meta-model, i.e., a Simulink model. While defining the steps, we had to address the following key questions:

How powerful in terms of the effect should a transformation step be?

The use of a powerful transformation step reduces the complexity of a refactoring specification but is more difficult to reuse. For instance, there are two possible ways to define the transformation step *deleteBlock* that deletes a block. One way is, if a block is removed, its incoming and outgoing segments remain and become unconnected segments. A more powerful version of *deleteBlock* would also remove the incoming and outgoing segments. The former is especially useful if after the deletion, the references to the now unconnected segments are still needed - if, for instance, the segments are rerouted to other blocks in a following transformation step. For the sake of reusability, we have decided to keep basic transformation steps as granular as possible. If necessary, more powerful versions are defined by composing more fine-grained steps, such as *deleteBlock-WithSignals*, which is realized by using *deleteBlock* and then deleting the incoming and outgoing segments using *deleteSegment*.

How can transformation steps be composed to define more complex transformation steps?

We distinguish between *elementary* and *composite* transformation steps. An elementary step modifies an instance of the meta-model without using other transformation steps, while a composite step consists of an ordered list of (possibly elementary or composite) child steps. Performing an elementary step directly changes an instance of the meta-model. A composite step can be performed by executing each step in the list in the specified order. Back to the previous example, *deleteBlock* is an elementary step while *deleteBlockWithSignals* is a composite step.

How should a step affect the layout?

Layouting of Simulink models ultimately addresses the positioning of blocks and signals in the Simulink Editor. Since the layout plays a crucial role for the readability of a Simulink diagram and layouting thus needs to be considered by refactoring operations, transformation steps can receive

```
Precondition: sameSubsystem(srcPort,targetPort)
              $\wedge \neg$ targetPort.hasIncomingSegment
1: function ADDSEGMENT(srcPort,targetPort, [name])
2: $p \leftarrow$ srcPort.containingSubsystem
3: newSeg $\leftarrow$ new Segment(name)
4: newSeg.source $\leftarrow$ srcPort
5: newSeg.target $\leftarrow$ targetPort
6: srcPort.outSegs $\leftarrow$ srcPort.outSegs $\cup$ {newSeg}
7: targetPort.inSeg $\leftarrow$ newSeg
8: p.childSegments $\leftarrow$ p.childSegments $\cup$ {newSeg}
9: end function
```

Figure 2. Algorithm for elementary transformation step *addSegment*

layout information from parent composite transformation steps. If layout information is not provided, predefined layout heuristics or Simulink itself determine the layout. For an improved layout after refactoring, the automatic layouting algorithm for Simulink diagrams [3] can be used.

Based on these basic principles, we show by examples how elementary and composite steps modify an instance of the meta-model. Figure 2 depicts (informal) pseudo code for the elementary step *addSegment* that adds a new segment from a source port *srcPort* to a target port *targetPort*. It presumes that *srcPort* and *targetPort* are in the same subsystem and *targetPort* does not have an incoming segment. If these conditions are satisfied, a new segment *newSeg* is created. A name is given depending on the type of refactoring in which this step is used. The segment's start and end ports are set to *srcPort* and *targetPort*, respectively. Additionally, the new segment is added to the collection of outgoing segments of *srcPort* and assigned to *targetPort* as the incoming segment. Finally, *newSeg* is added to the current subsystem.

Figure 3 shows the algorithm for *addCrossHierarchicalSignal* which adds (possibly cross-hierarchical) signals from a source port to one or several target ports. Unlike *addSegment*, *addCrossHierarchicalSignal* is a composite transformation step because it makes use of other transformation steps such as *addSegment*, *addOutportBlock* and *addInportBlock*. Note that the algorithm contains control structures, as well as other commands, and is not purely a list of transformation steps as indicated before. Due to space limitations, we abstract from implementation details here. As mentioned, the steps' algorithms as described in this paper are executed on an instance of the meta-model. Each call of an elementary transformation step is registered in an ordered step list which is then executed step by step on the real Simulink model.

The precondition of *addCrossHierarchicalSignal* states that the list of target ports must have at least one element and all target ports must be in the same subsystem. If a precondition is not satisfied, the entire refactoring in which this step is used will not be applied. If satisfied, it determines the subsystem where a forward constructed signal from source to target and a backwards constructed signal from target to source would meet (least common subsystem). Then, the signal is forwarded from the source

```
Precondition: sameSubsystem(targetPorts) $\wedge$ targetPorts $\neq \emptyset$
              $\wedge \forall$ tp $\in$ targetPorts: $\neg$tp.hasIncomingSegment
1: function ADDCROSSHIERARCHICALSIGNAL
                                        (srcPort,targetPorts)
2: leastSub $\leftarrow$ getLeastCommonSubsystem(srcPort,targetPorts)
3: curPort $\leftarrow$ srcPort
4: curSubsys $\leftarrow$ curPort.containingSubsystem
5: while curSubsys $\neq$ leastSub do
6:     outBlock = addOutportBlock(curSubsys)
7:     ret = addSegment(curPort,outBlock.inport)
8:     curPort $\leftarrow$ curSubsys.outportOf(outBlock)
9:     curSubsys $\leftarrow$ curPort.containingSubsystem
10: end while
11: subsysPath $\leftarrow$ getSubsystemPath(targetPorts(1),leastSub)
12: for p $\in$ sortByHierarchyTopDown(subsysPath) do
13:     inBlock = addInportBlock(p)
14:     addSegment(curPort,p.inportOf(inBlock))
15:     curPort $\leftarrow$ inBlock.outport
16: end for
17: for tp $\in$ targetPorts do
18:     addSegment(curPort,tp)
19: end for
20: end function
```

Figure 3. Algorithm for composite transformation step *addCrossHierarchicalSignal*

port up to the ancestor and from there down to the subsystem containing the target ports by creating outport blocks, inport blocks and signals for the intermediate subsystems. Finally, in the subsystem containing the target ports, branching signals are added from the newly added inport block to the target blocks. Note that *addCrossHierarchicalSignal* avoids redundant blocks and signals by creating a single signal path from the source port to the parent subsystem of the target ports before branching it to the target ports.

IV. SPECIFICATION OF REFACTORINGS

The transformation steps can be leveraged to formulate refactoring operations, as shown next using two examples: (1) *Replace Goto/From With Explicit Signals* creates explicit (possibly cross-hierarchical) signals from the source *Goto* block to all associated *From* blocks, (2) *Merge Subsystems* merges two subsystems into a single subsystem. These two refactorings are part of our Simulink refactoring catalog [2].

A. Replace Goto/From With Explicit Signals

Motivation: An advantage of data flow diagrams such as Simulink is that the data flow between blocks is explicit thanks to visual signal connections. However, Simulink provides *Goto/From* blocks as a means to define implicit, non-visual signal connections between blocks that may reside on different model levels - usually to reduce the visual complexity. Similar to the *Goto* construct in imperative programming languages, the use of *Goto/From* blocks, especially of global scope, may dramatically reduce the understandability of the model because tracing the data flow becomes more difficult. *Goto/From* blocks can be replaced by explicit signal connections without changing the behavior of the model. This can be a tedious task when done manually.

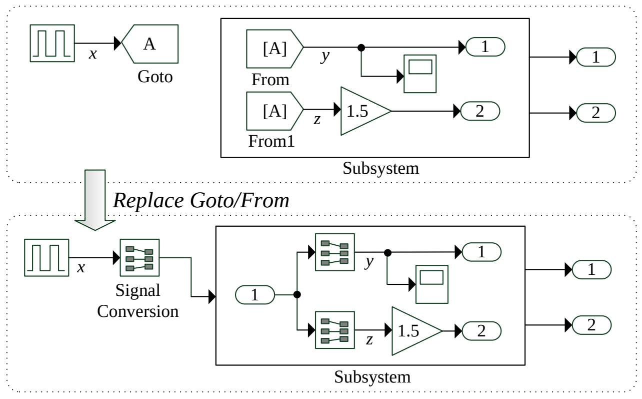


Figure 4. Example for refactoring *Replace Goto/From with Explicit Signals*

```
Precondition: -
 1: function ReplaceGotoFromWithExplicitSignals(gotoBlock)
 2: fromBlocks ← gotoBlock.fromBlocks
 3: inSeg ← gotoBlock.inSeg
 4: subsystemList ← buildSubsystemList(fromBlocks)
 5: gotoConverter ← replaceBlock(gotoBlock,'SignalConversion')
 6: curOutport ← gotoConverter.outport
 7: for s ∈ sortByHierarchyTopDown(subsystemList) do
 8:     targetPorts ← ∅
 9:     for fromBlocks ∈ s do
10:         for fromBlock ∈ fromBlocks do
11:             fromConverter ← replaceBlock(fromBlock,'SignalConversion')
12:             targetPorts ← targetPorts ∪ fromConverter.inport
13:         end for
14:     end for
15:     targetRootPort ← addCrossHierarchicalSignal(curOutport,targetPorts)
16:     curOutport ← targetRootPort
17: end for
18: end function
```

Figure 6. Algorithm for refactoring *Replace Goto/From With Explicit Signals*

Mechanics: Figure 6 shows (informal) pseudo code for *Replace Goto/From with Explicit Signal*. Take a look at Figure 4 for an example.

As first operation, the function *buildSubsystemList* is called to obtain a list of the corresponding *From* blocks' parent subsystems. This list is used for signal forwarding while avoiding redundant signal paths at the same time. Then, the transformation step *replaceBlock* is used to replace the *Goto* block with a *Signal Conversion* block. *Signal Conversion* blocks are used here solely for preserving signal names. More specifically, if the incoming signal of the *Goto* block has a different name than the signal names leaving the *From* blocks, the use of *Signal Conversion* blocks would allow these names to continue to exist after the refactoring.

Next in the algorithm, *subsystemList* is iterated. In each iteration, the *From* blocks within the current subsystem are replaced by *Signal Conversion* blocks. Finally, the composite transformation step *addCrossHierarchicalSignal* is used to foward signals to the *Signal Conversion* blocks.

B. Merge Subsystems

Motivation: During creation and maintenance of a Simulink model, reorganizing activities are frequent. In particular, it is often necessary to combine functionalities residing in separate subsystems into a single subsystem. With the current modeling support of the Simulink Editor, the modeler would have to cut and paste the content of one subsystem into the other subsystem. Then, the signals must be reconnected to re-establish the initial signal relationships. If lots of signals must be connected manually, this activity becomes both labor-intensive and error-prone.

Mechanics: While Figure 5 provides an example of this refactoring, Figure 7 shows (informal) pseudo code for merging two subsystems A and B. The precondition specifies that A and B must be non-atomic (virtual) subsystems. This restriction exists since merging atomic subsystems may change the behavior of the model.

In essence, the algorithm uses suitable transformation steps to move the content of B to A (line 18), adjust the signal connections (line 23, 26, 32, and 35) and finally delete B (line 37). Before the actual transformation, some book keeping needs to be done. In particular, *inSegsOfB* contains all incoming segments of B. Hash tables *inMap* and *outMap* are used to keep track of the references between inport and outport blocks of B to the source ports of the signals reaching them for reconnecting signals.

For *inMap*, if an inport block *inpBlock* of B has an incoming segment entering B at the inport corresponding to *inpBlock*, we store the mapping between *inpBlock* and the *source port* of that incoming segment returned by *getSrcPort*. In this context, the source port is returned by *getSrcPort*, which checks if the root source of the segment is an outport of A. If yes, we go into A and retrieve the source of the signal within A. Otherwise, the source port is the root source of the segment and is located on the common parent subsystem of A and B. For *outMap*, if an outport block *outBlock* of B has an outgoing segment leaving B at the outport corresponding to *outBlock*, we store the mapping between *outBlock* and the segment.

For transformation, *moveBlocks(B.content,A)* moves B's content to A. *inMap* is used to reestablish incoming signal connections to the blocks that used to be in B. In particular, for each inport block *inpBlock* stored in *inMap*, if the source port of *inpBlock* is in A, *inpBlock* is replaced by a *Signal Conversion* block before connecting the source port to the inport of that *Signal Conversion* block. Otherwise, the source port is connected to the inport of A corresponding to *inpBlock*. Then, using *outMap*, the initial outgoing segments of B are rerouted to the outports of A corresponding with the outport blocks being moved from B. Finally, the initial incoming segments of B and B itself are removed.

It should be noted that the decision of which of the two subsystems to be merged serve the role of A and B in the

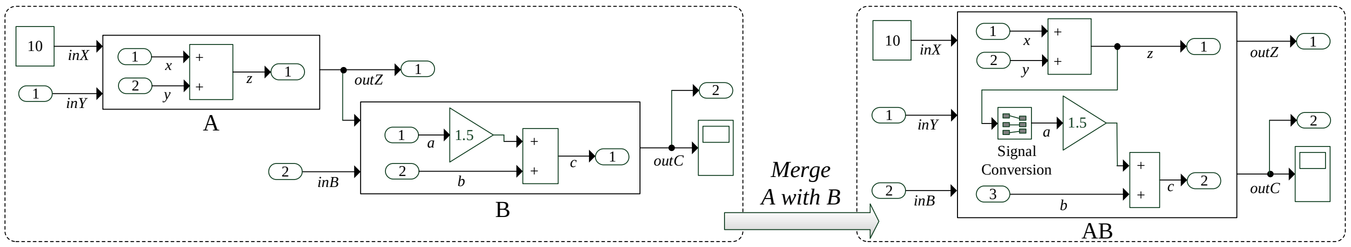


Figure 5. Example for refactoring *Merge Subsystems*

```
Precondition: $\neg$A.isAtomic $\wedge$ $\neg$B.isAtomic
function MergeSubsystems($A, B$)
  inSegsOfB $\leftarrow$ B.inSegments
  inpBlocksInB $\leftarrow$ B.inportBlocks
  for inpBlock $\in$ inpBlocksInB do
      inport $\leftarrow$ B.inportOf(inpBlock)
      if inport.hasIncomingSegment then
          inMap.value(inpBlock) $\leftarrow$ getSrcPort(inpBlock,A)
      end if
  end for
  outBlocksInB $\leftarrow$ B.outportBlocks
  for outBlock $\in$ outBlocksInB do
      outport $\leftarrow$ B.outportOf(outBlock)
      if outport.hasOutgoingSegment then
          outSeg $\leftarrow$ outport.outSegment
          outMap.value(outBlock) $\leftarrow$ outSeg
      end if
  end for
  moveBlocks(B.content, A)
  for inpBlock $\in$ inMap.keys do
      srcPort $\leftarrow$ inMap(inpBlock)
      if srcPort.containingSubsystem $=$ A then
          converter $\leftarrow$ replaceBlock
                                   (inpBlock,'Signal Conversion')
          addSegment(srcPort, converter.inport)
      else
          targetPort $\leftarrow$ A.inportOf(inpBlock)
          addSegment(srcPort,targetPort)
      end if
  end for
  for outBlock $\in$ outMap.keys do
      outport $\leftarrow$ A.outportOf(outBlock)
      outSeg $\leftarrow$ outMap.value(out)
      rerouteSegmentToNewSource(outSeg,outport)
  end for
  for seg $\in$ inSegsOfB do
      deleteSegment(seg)
  end for
  deleteBlocks(B)
end function
```

Figure 7. Algorithm for refactoring *Merge Subsystems*

algorithm affects the port order of the inports and outports within the resulting merged subsystem. This is due to the way Simulink automatically assigns port numbers when a port is added or deleted. However, since the port order does not affect the behavior, the refactoring does not change the model behavior. We have also defined a refactoring operation called *Reorder Ports* that can be used to rearrange the port order of inports or outports of a subsystem. If required, this refactoring can be used to achieve the desired port order.

V. Implementation

We have implemented a prototype in Matlab's *m* language that integrates refactoring support directly into Simulink Editor. Specifically, a refactoring operation can be directly triggered in Simulink Editor via a menu item or shortcut. Based on the *Template* design pattern [4], the prototype implements a generic workflow of refactorings as a graphical wizard. The behavior of a specific refactoring operation such as the required graphical dialogs for user input and the (interactive) specification of transformation steps can be easily defined and integrated into the prototype.

The meta-model in Section II is implemented as Matlab classes. The elementary and composite transformation steps in Section III are provided in the form of Matlab functions. In addition, the prototype also contains a collection of Matlab functions for model analysis that are useful for refactoring purposes. For instance, the functions *getLeast-CommonSubsystem* used from Figure 3 and *buildSubsystem-List* from Figure 6 are stored in a special collection since they are needed by multiple refactorings. The functions for transformation steps and model analysis serve as a high-level and compact API for formulating refactoring operations.

The prototype also features a graphical preview that shows the list of transformation steps to be executed in a tree. Moreover, it shows the Simulink diagram before and after a refactoring operation.

We have tested our prototype on several industrial Simulink models from the automotive domain at Daimler. The biggest time factor turned out to be the time required to convert a Simulink model into an instance of the meta-model. In an extreme case, for a model of about 20,000 blocks and a refactoring operation that affects almost the entire model, the parsing time took roughly 10 minutes. For most models and operations, however, the parsing time was just a matter of seconds. The transformation itself usually took only seconds, or at most, a few minutes.

VI. Related Work

In textual programming, refactoring has become a standard technique for restructuring code without changing its observable behavior [5], such as for object-oriented languages [5] and functional languages [6]. Modern Integrated Development Environments (IDEs) like Eclipse, NetBeans and Visual Studio offer built-in support for refactoring.

In model-based development, UML models have been targeted for refactoring support [7]. Refactoring of data flow diagrams such as Simulink, however, is only scantly researched. Sui et al. [8] propose an implementation approach for an automated refactoring tool aimed at visual dataflow programming languages. However, the focus of their paper is rather on the tool architecture aspect than on specifying refactoring operations modularly.

The current version of Simulink Editor does not provide refactoring support. Tools such as Model Advisor [9] or Model Examiner [10] can automatically detect violations of modeling guidelines and do provide, to a limited extent, so-called repair scripts for repairing guideline violations. Nonetheless, the focus of these tools is on automated detection of guideline violations and not on providing complex refactoring operations with possible user input or interaction.

The approach which is most related to our work has been developed in the MATE project [11]. It is an approach to visual specification and transformation for Simulink and Stateflow models based on graph transformation techniques. Specifically, modeling guideline violations and possible repair scripts are formulated in the graphical specification language called Story Driven Modeling (SDM). It turned out, however, that a purely visual specification language, such as SDM, is not powerful enough for complex real specification scenarios such as those including regular expressions, complex mathematical calculations and complex navigation through a network of linked objects.

VII. Conclusion and Future Work

In this paper, we have introduced our technique for specifying and implementing complex refactoring operations for Simulink diagrams based on the composition of transformation steps. The concept has been successfully implemented as a prototype that integrates refactoring support into Simulink Editor. Using the infrastructure provided by the prototype, we were able to implement many refactoring operations from our catalog [2] with little effort.

As the next step, we plan to extend our catalog and tool with further useful refactoring operations. Our future work will also address the automated identification of model constructs for which the application of certain refactorings is recommendable - so-called model smells, in analogy to code smells known from code refactoring [5]. There exist several techniques for Clone Detection in a Simulink diagram, as explored by Deissenboeck et al. [12] and Petersen [13], which could be used to identify similar or identical fragments in a Simulink diagram and suggest applicable refactoring operations for eliminating them. Moreover, we plan to evaluate the developed techniques and tool in real development environments at Daimler.

In addition, having automated transformation and refactoring techniques for Simulink models on hand, advanced applications are rendered possible. For instance, Simulink models could be automatically optimized by search-based algorithms using our transformation steps, as suggested for code [14], with respect to measurable model quality criteria, which already exist for Simulink diagrams [15].

References

[1] The MathWorks, "Matlab/Simulink," http://www.mathworks.de/products/simulink/ [Last access: 11/06/2013].

[2] Q. M. Tran and C. Dziobek, "An approach to design and maintenance of Simulink models by using transformations/refactorings and generation operations," in *Proceedings of the Model-Based Development of Embedded Systems Workshop (MBEES)*, 2013, pp. 1–12.

[3] L. Klauske and C. Dziobek, "Improving modeling usability: Automatic layouting for Simulink," in *Proceedings of the 2010 MathWorks Automotive Conference*, 2010, pp. 1–8.

[4] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[5] M. Fowler, *Refactoring: Improving the Design of Existing Code*. Addison-Wesley, 1999.

[6] H. Li, "Refactoring Haskell programs," Ph.D. dissertation, University of Kent, 2006.

[7] G. Sunyé, D. Pollet, Y. L. Traon, and J.-M. Jézéquel, "Refactoring UML models," in *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, 2001, pp. 134–148.

[8] Y. Y. Sui, J. Lin, and X. T. Zhang, "An automated refactoring tool for dataflow visual programming language," *SIGPLAN Notices*, vol. 43, no. 4, pp. 21–28, Apr. 2008.

[9] The MathWorks, "Model Advisor," http://www.mathworks.de/de/help/simulink/ug/consulting-the-model-advisor.html [Last access: 11/06/2013].

[10] Model Engineering Solutions, "Model Examiner," http://www.model-engineers.com/de/model-examiner.html [Last access: 11/06/2013].

[11] I. Stürmer, I. Kreuz, W. Schäfer, and A. Schürr, "The MATE approach: Enhanced Simulink and Stateflow model transformation," in *Proceedings of the 2007 MathWorks Automotive Conference*, 2007, pp. 1–9.

[12] F. Deissenboeck *et al.*, "Clone detection in automotive model-based development," in *Proceedings of the 30th International Conference on Software Engineering*, 2008, pp. 603–612.

[13] H. Petersen, "Clone detection in Matlab Simulink models," Master's thesis, Technical University of Denmark, DTU Informatics, 2012.

[14] M. Harman, S. A. Mansouri, and Y. Zhang, "Search-based software engineering: Trends, techniques and applications," *ACM Comput. Surv.*, vol. 45, no. 1, pp. 11:1–11:61, 2012.

[15] J. Scheible and H. Pohlheim, "Automated model quality rating of embedded systems," in *Proceedings of the 4th SQMB Workshop*, 2011, pp. 1–10.

Experiences on Mobile Cross-Platform Application Development Using PhoneGap

Jussi Ronkainen, Juho Eskeli, Timo Urhemaa, Kaisa Koskela-Huotari
VTT Technical Research Centre of Finland
Finland
jussi.ronkainen@vtt.fi, juho.eskeli@vtt.fi, timo.urhemaa@vtt.fi, kaisa.koskela-huotari@vtt.fi

Abstract—**Cross-platform mobile application development frameworks are an attractive alternative to native application development, with potential for improved asset reuse and reduced development costs. Few reports exist, however, on determining their suitability for a given type of application or identifying their potential pitfalls. To address this, we report our experiences from implementing a hybrid web application demonstrator on Android, iOS, Windows Phone 8, and desktop platforms for cloud-based content sharing and co-creation. The hybrid web application approach was found adequate for implementing the demonstrator. Notable challenges discovered during the process were platform dependent variation in HTML5 feature support, differences in the way browsers interact with platform services, and lack of platform specific debugging tools. Based on the results, emphasis on debugging tool support is suggested, as well as early and frequent testing on all target platforms.**

Keywords-cross platform; multi platform; phonegap; jquery; cordova; cloud; cloud-based; content; content sharing; liquid experience

I. INTRODUCTION

The current mobile device market is dominated by two operating systems (Q4 2012: Android 69.7%, iOS 20.9%), and the global smartphone sales for 2013 is estimated to be close to one billion units [1]. In this light, cross-platform development approaches, which facilitate application development for multiple operating systems with a single code base, seem compelling. Furthermore, in the current market situation there could be some room for a third competitor (e.g., Windows Phone or BlackBerry) into the mix of operating systems, which could make cross- platform mobile application development even more lucrative for software developers.

The advantages of a cross-platform development approach compared to a multi-platform approach using native development platforms come from the use of a single codebase, which in turn can result in improved asset reuse and reduced development and maintenance costs, for example. Additionally, the barrier of entry into mobile application development can be lower in cross platform development environments where HTML, CSS, and JavaScript technologies are commonplace [2].

The downside of the cross-platform mobile development approach is that it may not be suitable in all situations, for example when native look and feel in user interface is required, or in games where adequate performance cannot be guaranteed [2].

For the reasons mentioned above we wanted to study the feasibility of the cross-platform approach for a specific application, and to learn of the potential pitfalls with the approach. As a result we want to share the experiences gained to practitioners in the field in form of practices that did or did not work.

To achieve this, we implemented a hybrid web application demo for cloud-based content sharing and co-creation. Our aim was to study the practicalities of cross-platform development on the popular PhoneGap platform to gain an understanding of its strengths and weaknesses, as well as the skills and effort required. As a secondary objective, we studied the suitability of a hybrid web application approach for our particular application.

In the next section, the application concept is explained. Section 3 illustrates our implementation approach, along with expected results. Section 4 discusses mobile cross-platform development approaches with respect to identified state of the art. Results are described in Section 5, followed by conclusions and future work in Section 6.

II. CASE CONTEXT

The background for developing the application is in our previous research into the way people understand digital content, how they currently use it, and how they would want to use it. Sixty people participated in the research via the online user interaction forum Owela [3]. Of the 71 narratives and more than a thousand discussion comments provided by the participants, we chose photographs as the theme for our application.

We wanted to focus on the ease of content sharing and co-creation because of their perceived importance in many of the user stories. Cloud storage for the photos was also a recurring theme in the stories, and an evident requirement also because the wider context of the Cloud Sofware Program in which this research was carried out.

Our earlier research in content sharing and co-creation had also focused on the concept of liquid experience [4], which aims to provide users with a consistent experience regardless of the device used for accessing the information. This concept also gave us more freedom in choosing the cross-platform framework since following native application look and feel on each device platform was not deemed critical.

III. APPROACH

Wishing to experiment further with the liquid experience concept, we chose to implement the application as a hybrid web application that would allow running it standalone on Android, iOS, and Windows Phone 8 and, with some restrictions, on a desktop browser. For backend, we chose to use Google App Engine mostly because of our prior experience with it, and because it offers rudimentary image manipulation functionality we assumed could be useful. At a later stage in the project, we also evaluated the feasibility of porting the application to another backend, experiences of which will be briefly discussed later.

A. *Framework Selection*

We didn't want to limit the application to any particular platform or device. At the time of writing, HTML5 based approaches support the most platforms and also, via the use of CSS3, make adaptation to different screen sizes and orientations relatively simple. Of the available HTML5 cross-platform frameworks, we chose PhoneGap due to its widest device support and because it imposes minimal restrictions to applications that utilize it. Also, PhoneGap enables the packaging of HTML5 applications as native applications. PhoneGap ships without visual components, which makes it very flexible in terms of UI, but also means that the developer has to choose or implement all application components oneself and ensure they will work together. Browser based applications also have a performance overhead with respect to native applications but that was not considered an issue, since the performance requirements for our application were considered very modest.

Furthermore, the PhoneGap framework has a plugin interface for running native code that can access device capabilities. Many common plugins such as GPS, camera and local file access are implemented by default in PhoneGap. Utilizing these plugins does not require any native development skills. PhoneGap also supports custom plugins, so the application can be extended to use native code for functionality that is not supported in HTML5 or PhoneGap by default, or which would be computationally too intensive to implement in JavaScript. Although our application does not make much use of PhoneGap plugins, from a research perspective we found native code support to be an important feature in cross-platform development for added flexibility.

From application development point of view, there are many JavaScript frameworks available that focus on, for example, the graphical user interface and widgets, DOM tree manipulation, and web application architecture (Model-view-controller).

We chose the jQuery Mobile application framework as the JavaScript library for implementing the application UI. jQuery was used for DOM tree manipulation and Ajax based server requests. Our choice of frameworks was largely influenced by the vast popularity jQuery, and in case of jQuery Mobile, the fact that it seemed to provide wider platform support than most similar frameworks. Both jQuery Mobile and PhoneGap have active user communities and both projects are frequently updated and well documented. In addition, many examples and demos paved our way to choose these platforms.

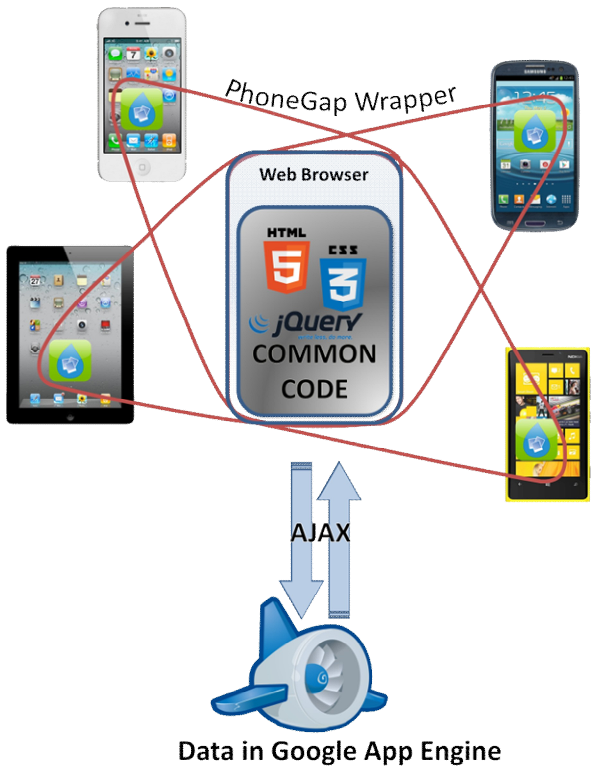


Figure 1. Our client/server structure.

B. *Development Methods and Tools*

We had three developers, each focusing on one platform in particular and the common codebase in general. This gave us an opportunity to observe the multi-platform development procedures with respect to, e.g., version control where the common application codebase had to be integrated into three platform specific codebases.

Our version control setup was such that there was a Git repository for each native project, and a repository for the common application code. The common repository was included into each native project as a Git submodule. Implementation was done on platform specific preferred editors for the native part - Xcode for iOS, Eclipse for Android, and Visual Studio for Windows Phone 8. HTML5, JavaScript and CSS3 editing was mostly done with JetBrains WebStorm.

The testing, largely UI driven and ad hoc, was done by the developers themselves on desktop browsers, mobile devices, and device emulators. Some functionality was also tested as automated unit tests on the Jasmine JavaScript unit test environment, which was run on a desktop browser.

C. *Expected Results*

From user interface point of view, we expected to have to make some conditional layout in the common code to cater for different screen sizes and resolutions. Also, we expected some minor variations in the way the application would render on the different devices. But for the most part, we assumed the ”write once, run anywhere” promise of cross-platform development to work more or less straight out of the box, especially since we were using the popular jQuery Mobile framework which we assumed to be well adapted for most platforms.

PhoneGap uses the device's browser as the application platform. Browsers are complex applications themselves, meaning that there is performance overhead for applications running on them. Also, since browsers have to handle all kinds of content, they are not optimized for any specific kind of application. Different browsers support HTML5 features to varying degrees, so application performance on different platforms might also vary. Performance was not, however, considered to be an issue in our case due to the simplicity of the application.

The use of HTML5 and jQuery / Ajax as the common implementation technology on all platforms was also expected to make cloud resource access simple.

IV. Related Work

We identified the current state of the art in mobile cross-platform development ([2][5][6][7][8][9]). In the following, cross-platform approaches are described in general, followed by a detailed discussion on one publication which most closely relates to our work discussed in more detail.

In *native application development* approach the application is implemented for a particular platform (as opposed to multiple platforms in cross-platform development) by using the provided Software Development Kit (SDK). The applications developed in this fashion maintain the look and feel of the platform. Porting the application to another platform is not possible without additional effort.

We consider *cross platform application development* to be development that is done with the help of cross-platform framework or with combination of platforms. Combining of platforms may be required because the frameworks focus on different purposes; some of them support development of complete applications that include application logic, user interface, and deployment, while some of them may focus on just one of these [2]. Related to UI representation in frameworks there are two different approaches commonly used; to imitate native look and feel (or use native components), or to maintain uniform look and feel for the supported platforms that ignores the native styling [2].

A definition of cross-platform frameworks is given by Sommer as follows: "Cross-platform frameworks are frameworks that support multiple platforms, with the same or similar effort involved to create an application on potentially more than one platform at once (or porting an application to other platforms with very little effort), as compared to creating it for only one platform with the native SDK. This essentially requires that a framework has to provide means to reuse parts of the architecture and source code that are platform-independent" [2].

The most commonly used frameworks in mobile cross-platform development can be categorized by the architectural approach taken into web-based, hybrid, and self-contained categories as presented in [2][6][7]. The publications also mention other types of approaches that utilize, e.g., cross compilation techniques. However, none of the current cross-compilation solutions that we are aware of are ready for production quality application deployments to prevalent mobile operating systems (e.g., Qt Alpha 5.1 advertises preliminary support for Android and iOS, with full support announced later in the oncoming 5.2 version).

By utilizing *web based frameworks* the application is developed as regular web site using HTML, CSS, and JavaScript technologies. An example framework in this category is jQuery Mobile. Pure web applications cannot be installed in similar fashion as native applications nor can they access the sensors or actuators of the mobile device.

In *hybrid frameworks,* the web based and native approach have been combined to create applications that inherit features of native applications (e.g., capability to install from an application store, native fashion application launching, capability to interface with sensors and actuators) but are developed using web technologies. An example framework from this category is PhoneGap.

Self-contained runtime environments, as described in [2], do not attempt to reuse existing web frameworks of the selected platform. By implementing their own web container the frameworks are in theory less constrained by any shortcomings in platform frameworks. Example of this type of framework is Titanium Mobile.

Zibula and Majchrzak [9] document the development of a Smart Metering Application using similar tool set as in our work. They outline the relevance of continuous testing on all target platforms because bugs might be visible only on a single platform. Our experiences also highlight the importance of continuous testing in cross-platform development. They also mention immaturity of the frameworks used, namely jQuery Mobile. We didn't face as severe problems in our work, which could be an indication that the frameworks have matured already. They also mention the debugging tools that they used, but don't go into detailed discussion about debugging, other than that the tools were very useful. Based on our experience debugging is one of the more important issues in cross-platform mobile development in which we focus in more detail in our work. Finally, they note that the hybrid approach is viable and advisable approach for cross platform development, but that in the long term it could be a transitional technology that may be replaced by pure HTML5 approach. While this may turn out to be true, we think that some form of tool or a solution is still needed to wrap the HTML5 application as a native application, and additionally HTML5 is unlikely to allow native extensions for whatever purpose. Zibula and Majchrzak also note that usability (of cross-platform developed mobile applications) and value for users are important research topics to consider besides technological development.

V. Results

Overall, we feel the application demo we implemented is complex enough to get an idea of the potential of hybrid web applications and to gather meaningful experiences from building it. Figure 2. shows a screenshot of the application during photo sharing on Android, Windows Phone and iOS devices. The figure illustrates differences due to the different fonts and screen aspect ratios on the devices.

We have divided our findings into three main groups; platform specific findings, user interface findings, and findings on the development process in general.

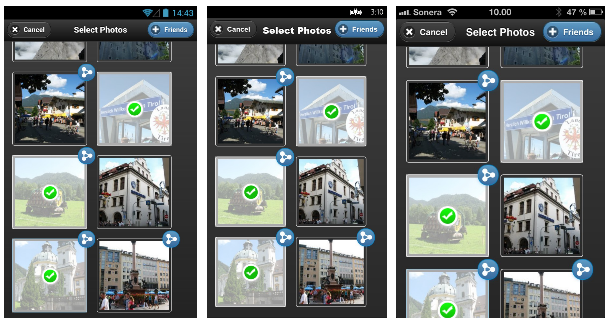


Figure 2. User photo sharing screen on Android (Galaxy Nexus, left), WP8 (Nokia Lumia 920, middle), and iOS (iPhone 4, right).

A. *User Interface Findings*

We found the user interface rendered from the common codebase to be fairly consistent among the platforms. This was largely due to our use of the jQuery Mobile framework which provided most of the UI elements. On the phones we tested, there were some nuances caused by different default fonts and different screen aspects, as illustrated in Figure 2. We used seven CSS3 media queries to set UI component dimensions to cater for all the screen sizes and orientations on the phones and tablets we had. In general, we found the underlying browser engines to do a good job in laying out the application on different screens and orientations. Some layout issues were discovered, such as different default page footer element handling on WebKit based vs. Windows Phone 8 devices but these could be fixed with platform specific style definitions.

We also encountered a few UI issues that affected only some platforms, such as page transition animations flickering on Android and completely missing or visually different on Windows Phone 8, and difficulties in disabling the default visual cue when attempting to scroll past the end of page on Windows Phone 8. Some of these issues have already been fixed in recent jQuery Mobile and PhoneGap versions, and we assume such easily noticeable visual differences will be fixed in future versions. However, we had to use platform specific style definitions from time to time to enable, e.g., HW acceleration for UI transition effects.

Another source of UI issues was the virtual keyboard which is unique to each platform. The screen area taken by the keyboard varies, as does its interaction with the underlying application. In our experience, the effect of the virtual keyboard needs to be tested thoroughly on each platform.

Probably the most notable issue we discovered, however, was the occasional sluggishness of touch input. This seemed to affect all platforms at some time or another. Most commonly there were missed touch events such as pressing a button or starting a swipe. The issues were random and slight but still noticeable and detrimental to a smooth user experience. We did not analyze the cause of the sluggishness but to get the UI really responsive would probably require platform specific analysis and optimization of the HTML5/JavaScript/CSS3 code. Also, we did not pay any attention to DOM tree optimization, which at least in large applications could have a significant effect in application performance.

In general, UI event support was found to differ between browsers and if mobile and desktop browsers are to be supported, both touch and mouse events need to be handled. Also, touch event support differs between platforms – for example, not all jQuery Mobile swipe events work on Windows Phone 8 without platform specific HTML5 style definitions. For this reason it is necessary to test all UI events as early as possible on all devices, and support multiple navigation methods where possible.

B. *Platform Specific Findings*

In addition to user interface issues which were caused by the differences in browser rendering engines, there were a couple of platform specific issues we could not solve or circumvent by modifying the application.

By request from a Cloud Software Program partner, we briefly experimented with the possibility of porting the application to use another backend. During our trials with the second backend which used HTTPS we came across a problem with SSL certificates. The development installation of the backend used a static IP address without a domain name, which meant that browsers could not ensure the authenticity of the certificate. On desktop browsers we could add an exception, and on the Android PhoneGap version we observed no issues. However, we could not get iPhone to create an exception for the server. This meant that the iOS application could not be run against that backend. While the problem is eliminated when the certificate is tied to a domain name, it could be a problem during development as in our case. Certificate handling was not tested using Windows Phone 8.

Another issue we could not solve from within the application was with browser cookies on Windows Phone 8. Our application uses a session cookie received from the server at login to identify the user during subsequent operations. PhoneGap obtains the cookie settings from the browser, but these settings vary between platforms. On Windows Phone 8, we had to change the system wide cookie settings manually on the browser of the device in order to get the application to store the session cookie. This, of course, is not acceptable for a consumer application. The need for cookies could be averted by implementing an authentication token scheme on the client and the server but that would require extra work.

Overall, however, we found cloud-based resource access straightforward and uniform across all platforms.

Native plugins are also a source of platform specific differences. It should be noted that even the plugins that ship with PhoneGap are not supported on all platforms, so the need for native support should be considered early on in a cross-platform development project. We implemented a native application settings screen on each platform and passed the settings to the HTML application via the plugin interface. Activation of the settings screen was also done via the interface. We found the plugin interface to work quite well. The native side of the plugins can be debugged on platform specific development environments like any native code.

C. *Development Method and Tool Findings*

JavaScript is an interpreted language, meaning that without a compiler, the role of the editor in finding programming errors is emphasized.

While all native development environments (Xcode, Eclipse, Visual Studio) support the development of HTML5 applications, none of them in our opinion match the best of dedicated HTML5 editors. Also, the use of a common editor for the HTML5 application by all developers in a project is justifiable in order to establish, e.g., common practices and file templates. While significant parts of an application can be implemented against a desktop web browser, deploying the application on a device, however, requires the native development environment. This causes extra steps and switching between applications in the development process.

We found automated unit testing useful in detecting problems in program logic earlier. Running unit tests with a framework such as Jasmine is quick and isolates program logic issues well. We ran a limited set of unit tests on a desktop browser and because of the ease of running the test suite, unit testing was useful in detecting programming errors quickly. Unit testing frameworks typically provide means for writing stubs, spies and mocks that enable the separation of, e.g., network code from the UI. This helps in isolating program logic issues and programming errors, but in our experience, automated unit testing frameworks are of limited use in exposing issues related to the target platform.

We also found the SW project structure to have significance in cross-platform development. Since in our case the common application code project was included as a subproject in each of the native projects, we occasionally ended up with subproject version conflicts. In the Git version control system the only links between the main repository and the submodules are submodule IDs which are saved in the main repository, and in some situations changes in the IDs are not automatically reflected into the submodules. As a result, we ended up cloning the common module as a separate project into the appropriate directory in each native project, and excluding the directory from version control in the native projects. Automatic refreshing of the subproject during native project refresh was thus lost, but in our case extra manual work caused by that was negligible since the native projects were changed much less frequently than the common project. Native project updates were mostly PhoneGap version updates. In our experience, however, they need to be done with care as PhoneGap version updates usually have to be synchronized between all native projects and the common project. Occasionally, a new PhoneGap version forced us to recreate the native projects from scratch. The documentation of the new release was also outdated at times, which caused some extra work to solve out the native project upgrade process.

To reduce the need for handling native projects, Adobe offers the cloud-based PhoneGap Build service which builds native applications from the HTML5, JavaScript and CSS code. There are, however, restrictions to custom plugins in PhoneGap Build.

The most significant shortcoming we experienced during development was the limited debugging ability of PhoneGap applications. The reason is that the embedded native browser PhoneGap uses is not accessible to a debugger on every platform, and thus problems that arise only on a specific platform may be very difficult to debug. At the time of writing, only BlackBerry and iOS browsers offer remote debugging that can be extended to PhoneGap applications. The Chrome browser on Android offers remote debugging but not via PhoneGap. Windows Phone 8 lacks remote debugging capability for both of the scenarios. At the time of writing, the best solution for remote debugging of hybrid web applications is Apple’s development tools for iOS. Xcode in combination with Safari on Mac offers all required debugging capabilities including DOM tree manipulation, breakpoints and variable inspector.

For most of the time we used a desktop browser for debugging, occasionally augmented by the PhoneGap Emulator on Google Chrome. The emulator was useful in verifying the UI with different screen sizes and resolutions, and getting a hang of using the native interfaces exposed by PhoneGap, although the emulator mostly uses mock data for them. A good rule of thumb for hybrid web application development is to use desktop browsers so that Chrome is used as a preliminary test for Android, Safari for iOS and IE for Windows Phone. Some browsers also have built-in tools for simulating different mobile device screen sizes.

Another useful PhoneGap debugging tool we used is weinre that is available either as a local installation or online via debug.phonegap.com. While weinre does not offer breakpoints, it does allow the inspection, highlighting and modification of DOM elements and JavaScript variables via a console.

PhoneGap can also relay the JavaScript console.log() output to the development environment console window. We found debug prints to console a viable debugging method, although understandably limited.

D. *Summary of Findings*

HTML5-based cross-platform applications rely heavily on the web browser on each platform, and differences in how the browsers implement HTML5 features were the underlying cause for most of our findings. In particular, we found occasional platform specific issues with page element layout and certain jQuery Mobile page animations, and touch event support. Most issues were solved by platform specific code and style definitions, but the intermittent problems with touch input responsiveness on all platforms were not.

Issues were also encountered in the way the browsers interact with their surroundings, namely in the visual cue the browsers give on trying to scroll past page boundaries, virtual keyboard behaviour, SSL certificate handling, cookie handling, and PhoneGap plugin support. While some of the issues were remedied via native project settings, solutions were not found during this study for the SSL certificate and cookie problems.

From a developer viewpoint, we found a dedicated HTML editor more useful than native IDEs which are typically not optimized for editing HTML5. Support for debugging on the device is only possible on iOS and Blackberry at the moment, which was found to be the biggest drawback of the approach. When device debugging is not required, desktop browsers provide good debugging options – although their use is not as seamless as debuggers on native IDEs.

VI. Discussion

In our experiment, we implemented a content sharing and co-creation application using PhoneGap and jQuery Mobile. We found the approach to fit our type of application well, and platform specific additions to the common codebase to be fairly minimal. HTML5 and CSS3 were found to do an efficient job of scaling the layout to different screen sizes and orientations, and that in general, the UI renders smoothly on the different platforms. However, we encountered issues with jQuery Mobile animations, so it is advisable to keep them to a minimum. This is particularly important if the targeted range of platforms is wide, or targeted devices are of modest performance or use old web browser engines.

There were also issues with UI responsiveness. Some issues we were able to fix via platform specific, non-standard style definitions, but we could not quite reach consistent, native quality responsiveness on any of the platforms.

Development tools were found adequate for most of the time, when the code could be developed and tested against a desktop browser. Automated unit testing was also experimented, and found useful in finding program logic bugs quickly.

Debugging on the target devices is the area that is in our experience most evidently lacking in hybrid web application development. The role of debugging is emphasized by the loosely typed, interpreted nature of Javascript, as without a compiler there are fewer safety nets to catch programming errors early. For limited device debugging we experimented with weinre and the PhoneGap emulator. Both were found useful, but lacking in functionality. Problems that do not surface on a desktop browser tend to concern non-standard HTML5 / CSS3 extensions or other platform specific browser behaviour. Thus, solving these problems is difficult without platform specific source-level debugging with breakpoints. For these reasons, the role of active and early testing on every platform is paramount.

VII. Conclusions and Future Work

The current smartphone and tablet market has made it necessary to develop applications for several platforms. Cross-platform development approaches are one way of increasing asset reuse between platforms and reducing development cost. Our study focused on the hybrid web application approach using the popular PhoneGap platform.

Overall, the approach was found solid and suitable for the type of application presented in the study. The biggest drawback encountered in the approach is insufficient debugging support on mobile devices. Platform specific variation in HTML5 feature support and browser interaction with the platform were found to necessitate constant testing on all platforms. UI performance issues that varied between mobile platforms were also encountered. Examining them would be one potential objective for future research.

Comparison of the hybrid web application approach with other cross-platform approaches would be another interesting topic, perhaps by implementing the same demonstrator using different approaches.

Acknowledgment

The authors wish to thank the TiViT Cloud Software Program in which this research has been carried out.

References

[1] Gartner, “Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012”, Press release, http://www.gartner.com/newsroom/id/2335616 08.08.2013

[2] H. Heitkötter, S. Hanschke, and T. A. Majchrzak, “Evaluating Cross-Platform Development Approaches for Mobile Applications,” in Lecture Notes in Business Information Processing, Volume 140, 2013, pp. 120-138

[3] P. Näkki and K. Koskela-Huotari, “User Participation in Software Design via Social Media: Experiences from a Case Study with Consumers,” in AIS Transactions on Human-Computer Interaction, vol. 4, 2012, pp. 128-151.

[4] H. Kiljander and V. Nore, “Experiences from Long-Term Online User Collaboration in Strategic Product Design,” in Proceedings of NordiCHI 2012, Industrial Track, ACM.

[5] A. Sommer, Comparison and evaluation of cross-platform frameworks for the development of mobile business applications, Master’s thesis, Fakultät für Informatik, Technische Universität München, 2012.

[6] A. Holzinger, P. Treitler, and W. Slany, “Making Apps Useable on Multiple Different Mobile Platform: On Interoperability for Business Application Development on Smartphones,” in Multidisciplinary Research and Practive for Information Systems, Lecture Notes in Computer Science, Volume 7465, 2012, pp. 176-189.

[7] E. Masi, G. Cantone, M. Mastrofini, G. Calavaro, and P. Subiaco, ”Mobile Apps Development: A Framework for Technology Decision Making,” in Mobile Computing, Applications, and Services. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, Volume 110, 2013, pp. 64-79.

[8] L. Corral, A. Janes, and T. Remencius, “Potential Advantages and Disadvantages of Multiplatform Development Frameworks – A Vision on Mobile Environments,” in Procedia Computer Science, Volume 10, 2012, pp. 1202-1207.

[9] A. Zibula and T. A. Majchrzak, “Cross-Platform Development Using HTML5, jQuery Mobile, and PhoneGap: Realizing a Smart Meter Application,” in Web Information Systems and Technologies, Lecture Notes in Business Information Processing, Volume 140, 2013, pp. 16-33.

A Real-Time Design Pattern for Actuators in Advanced Driver Assistance Systems

Hela Marouane*, Achraf Makni†, Claude Duvallet*, Bruno Sadeg* and Rafik Bouaziz†
*University of Le Havre
Le Havre, France
Email: {Hela.Marouane, Claude.Duvallet, Bruno.Sadeg}@litislab.fr
†University of Sfax
Sfax, Tunisia
Email: {Achraf.Makni, Raf.Bouaziz}@fseg.rnu.tn

***Abstract*—Advanced Driver Assistance Systems are hard real-time control systems in the automotive domain. They consist mainly of data acquisition, decision and action subsystems. The action subsystem constitutes a complex system which is composed of several embedded devices. The design of these systems is considered to be a complex process, as all components and real time constraints have to be considered during the design. Failures in hard systems could result critical situations. To tackle this problem, the design patterns present a reuse solution that improves the quality of the development process and reduces the complexity of systems design. However, the patterns which exist in the literature are abstract and do not represent the advanced driver assistance systems. In this paper, we focus on defining a specific real-time design pattern for an action subsystem of an advanced driver assistance system. This pattern captures the structural and the behavioral aspects. The definition of this pattern is based on a development process. To make this pattern more flexible and understandable, we add some semantics to the UML concepts using an UML-Profile, which expresses the real-time elements of the pattern and its variability.**

Keywords*—*Design pattern; Real-Time; UML-Profile; Actuator; ADAS.

I. Introduction

In recent years, the number of vehicles on the road has greatly increased. To reduce the risk of accidents, new technologies in vehicles, called Advanced Driver Assistance Systems (ADAS), have been appeared. Among these systems, we can quote Adaptive Cruise Control (ACC) [1] and Lane Departure Warning system [2]. Furthermore, ADAS systems help drivers in their driving tasks. As a result, the use of these systems improves road safety and reduces the risk of accidents. An ADAS is a complex real-time (RT) embedded system which consists of three subsystems:

1) The data acquisition subsystem: it includes a series of sensors (e.g., radar and wheel speed sensors) and a sensor data fusion unit that allows computing appropriate sensors data to estimate the consistent state of a vehicle and its environment [3].
2) The decision subsystem: it uses the data fusion unit outputs to analyze the current situation and decide the appropriate actions to be transmitted to actuators [4].
3) The action subsystem: it reacts to the decision subsystem by (i) providing automatic actions such as braking, and/or (ii) delivering visual, acoustic or haptic warning information to the driver [5]. This subsystem is consisting of several technologies (e.g., automatic actuators and Human Machine Interface), which serve more sophisticated functions.

The design of ADAS is highly complex; it is difficult to model the components, their interactions and the time constraints related to both data and transactions. Most often the accidents that are caused by failing developed systems, due the errors in the design phase. Moreover, the interaction with a human driver introduces even more complexity, since a driver can behave unpredictably to warnings or automatic action. This problem adds a level of complexity to the design of these systems. In addition, ADAS may be implementing with different platforms, but implementation details and design methods are absent. For these reasons, it is essential to capture the design into appropriate methods that can be analyzed and applied to each system.

A way to design these systems may be to exploit reusable components like design patterns. These patterns provide abstract components that aim to facilitate systems design, leading to efficient and reuse solutions. Design patterns can be classified into general or domain-specific. General patterns are intended for several domains; so, they are often too abstract [6]. The problem with this category of patterns is to determine in which context or in which part of the system they can be applied. On the other side, domain specific design patterns often provide an optimal solution for a particular domain. In fact, they provide to the designers some well-defined concepts (e.g., attributes and methods) of the domain. For these reasons, several works [7][8][9][10] have proposed domain specific design patterns applied to the RT domain.

In this paper, we are interested to define a real-time design pattern that models the ADAS action subsystem, which is one of the complex subsystems composing an ADAS. This pattern models the structural and behavioral aspects in common way of an ADAS action subsystem which may be implemented with different languages and tools. The proposed pattern omits sufficient description of device characteristics, control algorithms, user interface and mechanical actuators design and alerts generation to construct an ADAS correctly. Moreover, this pattern allows designers to build an ADAS system without starting from scratch since the pattern models the common concepts of ADSA systems. To define this pattern, we apply a development process composed of three steps: (i) The study and the modeling of several representative real

ADAS in order to highlight their similarities and differences. (ii) The identification of the concepts of these systems, their similarities and differences to define our pattern. (iii) The application of a set of rules defined by Rekhis *et al.* [11], with some adaptations. Indeed, we have adapted some of them to ADAS systems and we have added some others for class and sequence diagrams. In addition, we have added some semantics to some basic UML concepts to make this pattern more flexible. This semantics consists of applying the UML-RTDB2 profile [9] that contains a set of stereotypes which express timing constraints, non functional properties and the variability of the pattern.

II. Related work

RT design patterns are reusable components that can be applied in the design phase in order to reduce the complexity of the software design. For this reason, several works [8][12][13][14][15][16] have defined RT design patterns. Among these works, (i) Slutej *et al.* [14] have proposed design patterns which model the real-time components behavior of an industrial turntable system using state machine diagrams, (ii) Konrad *et al*. have proposed in [16] patterns to model structural and behavioral parts of embedded systems. These patterns are applied to applications from automotive domain. These patterns do not describe all specificities of an action subsystem of an advanced driver assistance system; the designer must add the specific components, attributes and operations of ADAS action subsystem. Therefore, the system can be developed with anomalies, and (iii) Armoush *et al.* have defined in [7] a template of design patterns which aim at modeling safety-critical embedded systems. This template shows the implications of the patterns on the non-functional requirements including safety, reliability, modifiability, cost and execution time. These patterns do not represent the functional aspects and the architecture of an embedded system. These patterns do not take into account the time constraints related to both data and transactions. For these reasons, we are interested to model RT design patterns that take into account these requirements.

Rekhis *et al.* [10][12] have proposed RT domain specific design patterns which model RT data acquisition and decision subsystems. These patterns allow modeling the structural and behavioral aspects for these subsystems. In [10], we find a RT design pattern which models the decision subsystem of RT applications that need to be managed by database systems. In addition, Rekhis *et al.* [12] have proposed RT design patterns which model the RT data acquisition. They describe how to model the requirements (real-time data and real-time transactions) and non functional aspects of RT applications. They have also defined another RT design pattern which models the multi-versions RT data which allow maintaining for each data item related to a measure type (e.g., velocity and position) multiple versions in order to reduce data access conflicts between transactions [12]. In addition, they express the variability of the patterns to facilitate and guide their reuse. We agree that the expression of the non functional requirements and the variability are very important for the design of RT applications. In fact, the variability is an important criterion to maximize pattern reuse, and the non functional aspects play an important role in the quality of the development process. So, we will take into account these aspects to model our pattern. However, the patterns presented in [10] and [12] are at a high abstraction level; they do not clearly differentiate between some concepts of real-time applications, such as the sensor and derived data. Thus, the patterns instantiation is complex and the developed system cannot meet all its requirements; the designer must identify and model the entities, their attributes, their relationships and their operations, that are not showed in the pattern according to a specific RT application. Moreover, these patterns describe the RT domain in general. They do not clearly represent some time constraints like the deadlines of actions. Modeling time constraints is very important since once these constraints are taken into account, they can help to verify and understand the temporal behavior and aid in the development of RT systems. When RT constraints are not satisfied (e.g., missing of the transaction deadlines), it can result in a system failure. For these reasons, we define our RT design pattern which takes into account these constraints.

However, to the best of our knowledge, there are no patterns exist in the literature to model the action subsystem (actuators and HMI devices). For these reasons, we define a new RT design pattern, named ADAS-Action Subsystem (ADAS-AS), which takes into account the specific constraints and requirements related to the action subsystem of ADAS. This subsystem is responsible for handling the outputs from all the different applications in order to carry out appropriate intervention strategies (automatic actions and warnings) to reduce critical situations.

III. UML-RTDB2 profile stereotypes

In this section, we describe the stereotypes of UML-RTDB2 profile we have proposed in [9]. This profile is an extension of UML 2.1.2 [17] to represent real-time characteristics of ADAS systems. It provides features to express (a) the variability of the patterns, (b) the real-time constraints and (c) the non functional properties.

The variability of patterns is an important criterion to obtain a flexible pattern. To specify the variability of patterns, we have used the following stereotypes [18] to extend the class diagram of our pattern: (a) $<< mandatory >>$ which specifies the fundamental classes and relations that must be instantiated when the model is applied to a specific application, (b) $<<$ *optional* $>>$ which is used to express optional features (e.g., classes, attributes, operations and relations). The optional element can be omitted in a pattern instance and (c) $<< extensible >>$ which indicates that a concerned class in a model may be extended by adding new attributes and/or methods during pattern reuse. This stereotype has the following tagged values: *extensibleAttribute* and *extensibleMethod* which are boolean. With true value, they indicate that the model can be extended by adding new attributes (if *extensibleAttribute* is true) and new methods (if *extensibleMethod* is true) in a pattern instance. We extend also our pattern sequence diagram using the stereotypes $<< mandatory >>$ and $<<$ *optional* $>>$ which are applied to the interaction fragments, lifelines and messages.

In order to model the RT features of ADAS, we have also imported some stereotypes from UML-RTDB [19] and from NFP (Non Functional Properties) sub-profile of MARTE [20]. From UML-RTDB, we have imported the following stereotypes: (a) $<< sensor >>$ which is applied to a class interface and indicates that the measurement is a sensor data, (b) $<< derived >>$ which is applied to classes and is used to express derived data that are calculated from sensor data, and (c) $<< periodic >>$ and $<< sporadic >>$ which are applied to express periodic and sporadic methods, respectively. The $<< periodic >>$ stereotype is characterized by a deadline and a period. The $<< sporadic >>$ stereotype is characterized by a deadline and a triggered time. From NFP sub-profile of MARTE, we have imported the following stereotypes: (a) $<< nfp >>$ that declares non functional requirements and (b) $<< nfptype >>$ that extends the DataType metaclass. It is used to specify NFP values such as *NFP_Duration* and *NFP_Frequency*. In addition, we have expressed real-time constraints with OCL (Object Constraint Language) [21].

IV. Development process for the RT design pattern

In this section, we propose a development process to define a RT design pattern in order to facilitate the design of ADAS applications. To be able to define this pattern, we study and model several ADAS systems in order to determine each application model (i.e., class and sequence diagrams). These models allow extracting the similarities and differences which are represented using class diagram and sequence diagram. The identification of similarities is based on a semantic comparison between different concepts through a domain dictionary. This dictionary holds for each term the synonyms, the variations and the hyponyms. The common concepts are added to the pattern as fundamental elements whereas the different concepts are added as optional elements. The defined patterns are applied to model each ADAS system in order to validate them. The quality of these patterns is evaluated through amount of reuse metrics [22].

In order to derive the pattern class diagram, firstly, we adopt and adapt a set of rules defined in [11]. It is proposed in [11] to represent a fundamental class with a highlighted border and an optional class with a simple border. These representations have not added semantics to the model. For this, we propose to use the following stereotypes to add semantics and make the pattern more flexible and understandable: (i) $<< mandatory >>$ for fundamental classes and (ii) $<< optional >>$ for the optional classes.

Then, we add some rules, which are not defined in [11]. These rules are expressed through the following relations [23]:

- ○ *N_var(C_{A1},...,C_{An})* means that the names of the classes are a variation of a concept such as proprioceptive sensor and exteroceptive sensor.
- ○ *Att_equiv(C_{A1},...,C_{An})* and *Op_equiv(C_{A1},...,C_{An})* means that the names of attributes and the names of operations respectively of classes are either identical or synonym.

The added rules are defined as follows:

- **RC-1:** If a class is present with variation names (*N_var(C_{A1},...,C_{An})*), but has equivalent attributes (*Att_equiv (C_{A1},...,C_{An})*) and operations (*Op_equiv(C_{A1},...,C_{An})*), then it is added as a fundamental class. The relations *N_var(C_{A1},...,C_{An})*, *Att_equiv(C_{A1},...,C_{An})* and *Op_equiv(C_{A1},...,C_{An})* are defined in [23]. We propose to use the stereotype $<< mandatory >>$ for this class.
- **RC-2:** If attributes (respectively operations) of a class, which is present in all applications, are present in several applications (in more than a fixed threshold (e.g., 50%) fixed by the designer), then they are added in the pattern as optional elements. We use the stereotype $<< optional >>$ for these elements.
- **RC-3:** If a relation exists between two mandatory classes, then it is added to the pattern as a fundamental relation and it is stereotyped $<< mandatory >>$. However, if the relation exists between two classes which one of them is optional, it is added to the pattern as an optional relation and it is stereotyped $<< optional >>$.

Rekhis *et al.* have defined in [11] some rules to derive the class diagram of the pattern, but they do not represent rules for sequence diagram. For this, we have proposed the rules to design the sequence diagram of the pattern. These rules are expressed using the following relations:

- ○ *N_equiv(O_{A1},...,O_{An})* means that the lifelines have identical or synonym names.
- ○ *N_dist(O_{A1},...,O_{An})* means that none of the above relations holds.
- ○ *N_equiv(M_{A1},...,M_{An})* means that the names of messages are either identical or synonym.

The proposed rules for sequence diagram are defined as follows:

- **RS-1:** If a lifeline is present in all applications with identical or synonym names (*N_equiv(O_{A1},...,O_{An})* [23]), then it is added to the pattern as a fundamental lifeline and it is stereotyped $<< mandatory >>$.
- **RS-2:** If a lifeline is present in several applications i.e., in more than a fixed threshold (e.g., 50%) fixed by the designer, then it is added to the pattern as an optional lifeline and it is stereotyped $<< optional >>$.
- **RS-3:** If a lifeline is too specific for an application (*N_dist(O_{A1},...,O_{An})* [23]), then it is not added to the pattern.
- **RS-4:** If the sender and the receiver are mandatory lifelines, and the message between them is present in all applications with identical or synonym names (*N_equiv(M_{A1},...,M_{An})* [23]), then it is added to the pattern as a fundamental message and it is stereotyped $<< mandatory >>$.
- **RS-5:** If the sender and the receiver are mandatory lifelines, and the message between them is present in several applications, then it is added to the pattern as an optional message and it is stereotyped $<< optional >>$.
- **RS-6:** If a message exists between two lifelines which one of them is optional, then it is added as an optional message and it is stereotyped $<< optional >>$.

- **RS-7:** If a combined fragment is present in all applications with synonym or identical names, it is added to the pattern as a fundamental fragment and it is stereotyped $<< mandatory >>$.

V. Building of an action subsystem pattern

In this section, we define a new specific real-time design pattern, entitled ADAS Action Subsystem (ADAS-AS), designed to model the architecture of ADAS actuators and HMI elements. The definition of the appropriate solution, in terms of static and dynamic views, is based on the process development described in the Section IV. In order to describe common and variable parts that must be present in the pattern, we begin to study and model three commercial ADAS systems among the systems which we have modeled. These systems are: Lateral Safe (LS) system that is representative of lateral control systems, Adaptive Cruise Control (ACC) system that is representative of longitudinal control systems and Saferider system that is representative of longitudinal and lateral control systems. These applications are designed by professors who have an experience in UML based on the study several documents provided by the automotive companies [5][1][24].

A. Description of ADAS systems

1) Lateral Safe system (LS): LS [5] is a system that reduces the risk of collisions in lateral and rear area of the vehicles. In addition, this system assists the driver in adverse or low visibility conditions. LS system warns the driver by using an effective HMI. This HMI has been evaluated and demonstrated in VOLVO cars [5]. LS system consists of several HMI elements: (i) The side and rear view mirrors HMI with leds which are activated in different colors and number, related to the danger level (e.g., cautionary and imminent warnings), (ii) the a-pillar with a symbol light, activated to warn the driver of the risk of a critical lateral collision and (iii) the car speaker, providing directional acoustical warnings in the case of imminent lateral collisions. The time warning depends on speed and driver reaction time and is presented to driver few times before the hazard using two warning levels (imminent danger and cautionary danger). The warnings are provided for a period with priority during each critical situation in order to reduce the number of false alarms. The HMI devices are activated via the HMI manager for each received action signals.

Figure 1 shows the class diagram which represents the HMI of LS system. This class diagram is resulted from the study of several documents provided by the automotive companies [5]. This model represents the following classes: (a) *HMIElement* class that contains the main properties of the HMI elements included in the lateral safe system; (b) *CarSpeaker* and *LEDDevice* that represent the subclasses of *HMIElement* generic class; (c) *HMIManager* class that activates the HMI warning elements; (d) *WarningSignalType* class that represents the type of warning provided by the HMI elements; (e) *BeepSound* and *LightSymbol* that represent subclasses of *WarningSignalType* class; (f) *WarningSignal* class that concerns the warnings delivred to the driver in critical situations; (g) *Driver* class that is associated with *WarningSignal* class to indicate that the driver will be warned in critical situations; (h) *Vehicle* class that is associated with *Driver* class to indicate that the driver has changed the status of the controlled vehicle taking into account the generated alert; (i) *DriverAction* class that represents the driver's reactions to the warning in order to avoid accidents. Figure 2 shows the sequence diagram which represents the dynamic aspect of the action subsystem of LS system.

2) Adaptive Cruise Control system (ACC): ACC system is an automotive application that is integrated and tested in modern luxury cars such as BMW [1]. ACC system aims at reducing the risk of accidents and providing safety and comfort to drivers and vehicles by adapting the vehicle's speed to the traffic environment. This system allows also keeping safe distance between the ACC-vehicle and the forward vehicle. The controller reads sensor data and calculates the desired acceleration or deceleration to maintain the safe distance. Then, it sends the corresponding values to the brake actuator or the throttle actuator. If a preceding slower vehicle is detected, ACC will decelerate the vehicle by applying the brakes (activate brake actuator) without driver application of the brake pedal to maintain a safe distance. In the absence of a preceding vehicle, ACC will accelerate the vehicle back to its set cruise control speed by activating the throttle actuator. In the case where braking is insufficient to maintain the safety distance, ACC will generate a light symbol and an audible distance alert if the intervention by the driver is needed to keep the safe distance.

Figure 3 shows the class diagram of the action subsystem of ACC system. This diagram represents the following classes: (a) *AutomaticActuator* class that contains the main properties of the automatic actuators of ACC system. *AutomaticActuator* class concerns each device in a car that executes some kinds of automated mechanical actions such as brake and throttle devices, (b) *DashboardDisplay* that includes the main properties of the HMI element, (c) *BrakeActuator* and *ThrottleActuator* that represent the subclasses of *AutomaticActuator* generic class, (d) *AutomaticAction* class that models the actions triggered by automatic actuators; (e) *InterfaceActuator* class that properly activates the HMI components or the mechanical actuators; (f) *WarningAlarm* class; (g) *WarningSignalType* class that constitutes the type of signals; (h) *BeepAlert* and *SymbolLight* that represent subclasses of *WarningSignalType* class; (i) *Driver* class and *CorrectiveAction* class; (j) *Car* class that is associated with *Driver* and *CorrectiveAction* classes to indicate that the driver modifies the status of the vehicle taking into account the warning signal. Besides, *Car* class is associated with *AutomaticAction* class to indicate that the status of the vehicle can be updated by activating the brake or the throttle actuators. Figure 4 presents the sequence diagram of the actuators and the HMI elements of ACC system to model the interactions between the components of the action subsystem.

3) Saferider system: Saferider system (www.saferider-eu.org) is an advanced telematics for enhancing the safety and comfort of motorcycle riders [24].

It consists of the following functions: (a) speed alert that alerts the rider when the speed exceeds the legal speed limits, (b) curve speed warning that alerts the rider when his/her speed is too high into a curve, (c) frontal collision warning that warns the rider when an obstacle is detected in front of the motorcycle and (d) intersection support that alerts the rider when a danger is present in intersections.

These functions are based on the comparison between

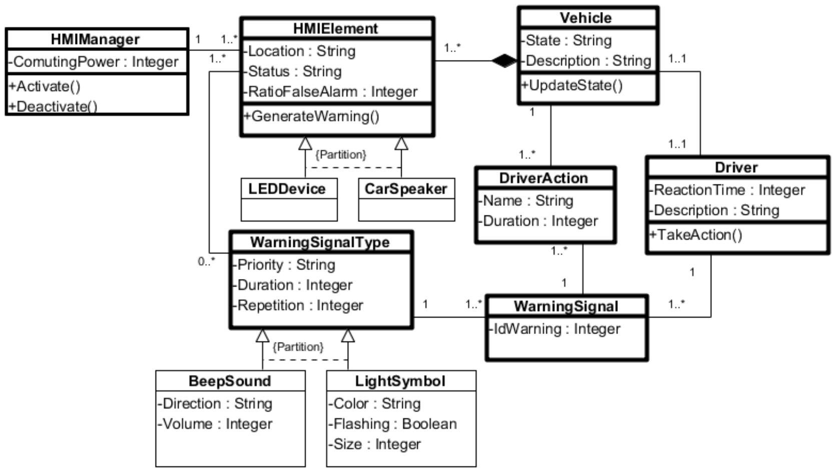


Fig. 1. Class diagram of the action subsystem of LS system.

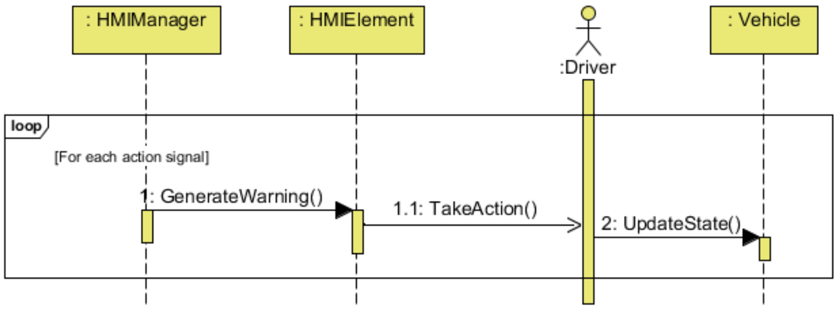


Fig. 2. Sequence diagram of the action subsystem of LS system.

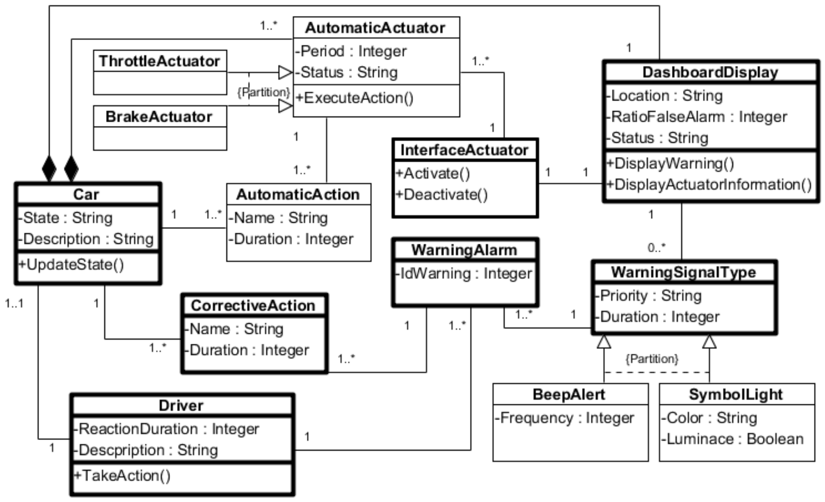


Fig. 3. Class diagram of the action subsystem of ACC system.

the actual rider manœuvre and the safe reference manœuvre which is calculated based on both the motorcycle's dynamics and the road characteristics. Once a hazard is detected, the warnings are generated through the following HMI elements which are activated using the HMI manager: (i) Head Up display integrated in the helmet, dashboard display and visual attractor on rear mirror providing visual warnings, (ii) in-helmet speakers providing audio warnings (e.g., acoustic and speech messages), (iii) haptic seat, haptic throttle, haptic golve and haptic handle providing haptic warnings (i.e., vibration). The HMI elements have been tested and demonstrated on the Yamaha and the Piaggio [24][25]. Figure 5 and Figure 6 illustrate, respectively, the class diagram and the sequence diagram of the HMI of Saferider system.

We note that the common elements are represented with a bold lines in Figures 1, 3 and 5.

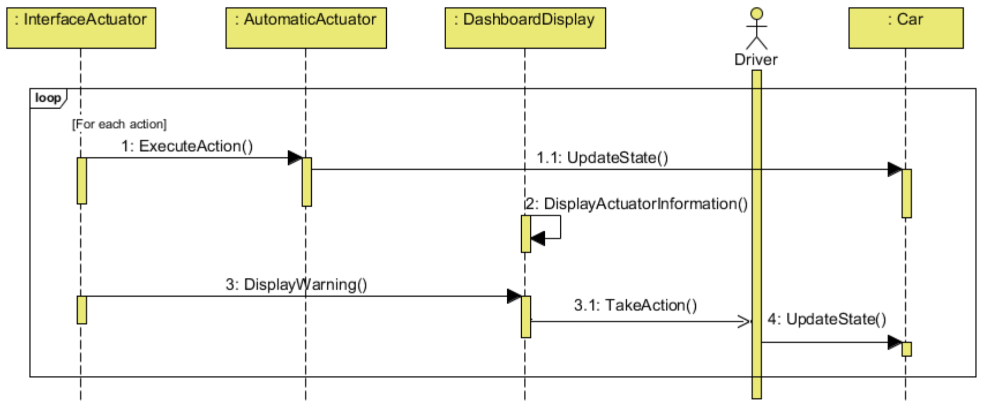


Fig. 4. Sequence diagram of the action subsystem of ACC system.

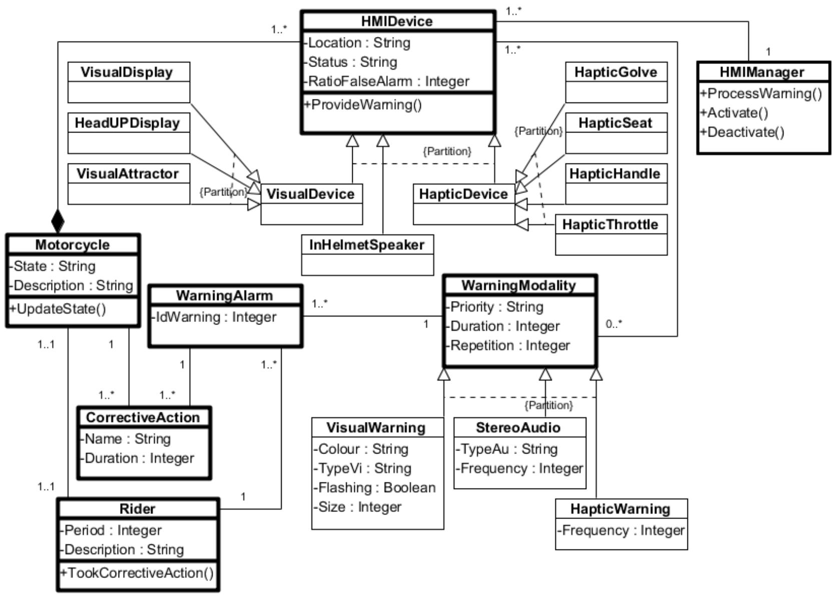


Fig. 5. Class diagram of the action subsystem of Saferider system.

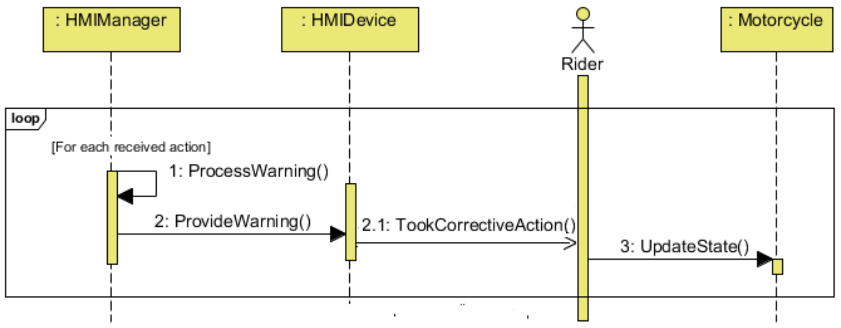


Fig. 6. Sequence diagram of the action subsystem of Saferider system.

B. Application of rules to define ADAS-AS pattern

We note similarities and differences between LS system, ACC system and Saferider system. The identification of these similarities and differences allows us designing the pattern class and sequence diagrams. The design of our pattern class diagram is based on applying the unification rules as shown in the following:

- The following elements are equivalent:
  - *HMIElement (LS)*, *DashboardDispaly (ACC)* and *HMIDevice (Saferider)*.
  - *HMIManager (LS)*, *InterfaceActuator (ACC)* and *HMIManager (Saferider)*.
  - *Vehicle (LS)*, *Car (ACC)* and *Motorcycle (Saferider)*.

- ◦ *Driver (LS)*, *Driver (ACC)* and *Rider (Saferider)*.
  - ◦ *WarningSignal (LS)*, *WarningAlarm (ACC)* and *WarningSignal (Saferider)*.
  - ◦ *WarningSignalType (LS)*, *WarningSignalType (ACC)* and *WarningModality (Saferider)*.

  HMIElement, *Vehicle*, *Manager*, *Driver*, *WarningSignal* and *WarningSignalType* classes are added to the pattern as fundamental classes and they are stereotyped $<< mandatory >>$.

- Visual, audio and haptic warnings (*N_var(BeepSound (LS), LightSymbol (LS), VisualWarning (Saferider), StereoAudio (Saferider), HapticWarning (Saferider), BeepAlert (ACC), SymbolLight (ACC))* [23]) are variable elements (i.e., represented with specialization relationships). Thus, *VisualWarning*, *AudioWarning* and *HapticWarning* are added as optional classes and they are stereotyped $<< optional >>$.

- *AutomaticAction* and *AutomaticActuator* classes are present in ACC system and in other modeled systems which exist in the literature. Thus, *AutomaticAction* and *AutomaticActuator* are added to the pattern as optianl classes and they are stereotyped $<< optional >>$.

- *LEDDevice*, *CarSpeaker*, *ThrottleActuator*, *BrakeActuator*, *VisualDevice*, *VisualDisplay*, *HeadUpDisplay*, *VisualAttractor*, *InHelmetSpeaker*, *HapticDevice*, *HapticThrottle*, *HapticHandle*, *HapticSeat* and *HapticGolve* are specific classes for each application. Thus, they are not added in the pattern.

- Once the classes are added to the pattern, we define the relations between classes. For example, the system provides warnings to the driver in critical situations. Thus, it exists a relation between *Driver* and *WarningSignal* classes. That is, *Driver* class is associated with the *WarningSignal* class. *Driver* and *WarningSignal* are mandatory classes, thus the association between them is added to the pattern as a fundamental relation which is stereotyped $<< mandatory >>$.

The design of our pattern sequence diagram is based on the application of the unification rules (Section IV), as shown in the following:

- The following elements are equivalent:
  - ◦ *HMIManger (LS)*, *InterfaceActuator (ACC)*, *HMIManager (Saferider)*.
  - ◦ *HMIElement (LS)*, *DashboardDisplay (ACC)*, *HMIDevice (Saferider)*.
  - ◦ *Driver (LS)*, *Driver (ACC)*, *Rider (Saferider)*.
  - ◦ *Vehicle (LS)*, *Car (ACC)*, *Motorcycle (Saferider)*.

  Rule RS-1 is applied by adding the following lifelines: *Manager*, *HMIElement*, *Driver* and *Vehicle* to the pattern sequence diagram as fundamental elements and they are stereotyped $<< mandatory >>$.

- *AutomaticActuator* lifeline is present in several applications (ACC and other modeled systems). Rule RS-2 is applied by adding this lifeline to the pattern sequence diagram as an optional element and it is seterotyped $<< optional >>$.

- The following messages are equivalent:
  - ◦ *GenerateWarning() (LS)*, *DisplayWarning() (ACC)*, *ProvideWarning() (Saferider)*.
  - ◦ *TakeAction() (LS)*, *TakeAction() (ACC)*, *TookCorrectiveAction() (Saferider)*.
  - ◦ *UpdateState() (LS)*, *UpdateState() (ACC)*, *UpdateState() (Saferider)*.

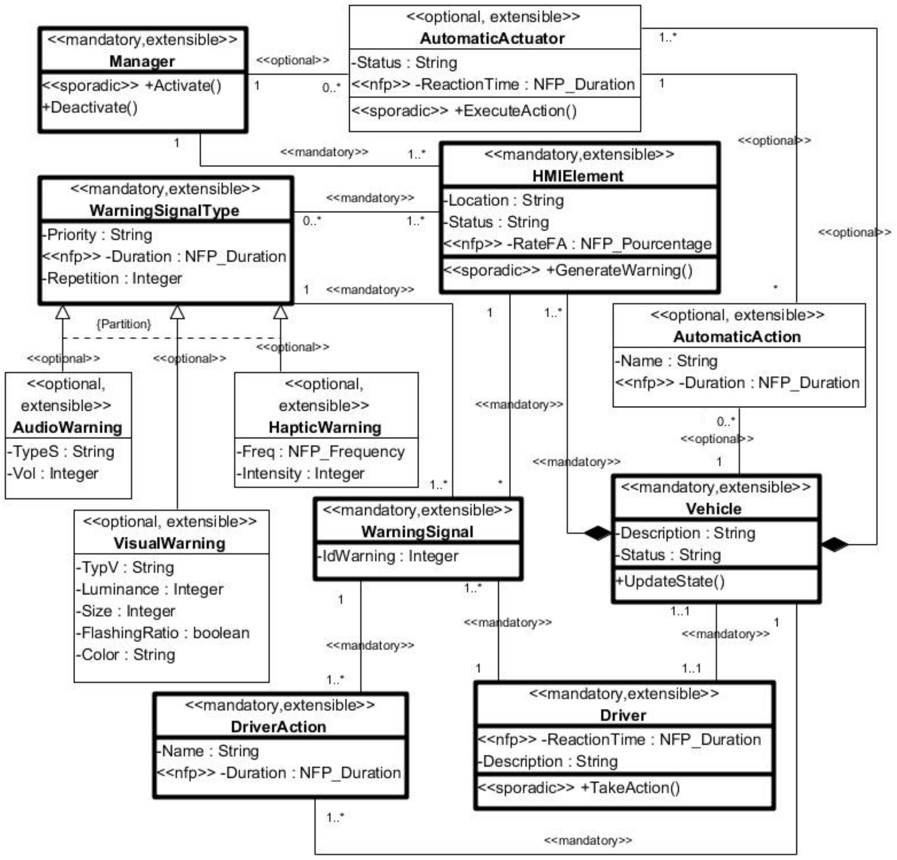
  Rule RS-4 is applied by adding *GenerateWarning()*, *TakeAction()* and *UpdateState()* messages to the sequence diagram as fundamental elements.

- *ExecuteAction()* is present in ACC system and other modeled systems. It exists between *InterfaceActuator* and *AutomaticActuator*. Rule RS-6 is applied. *ExecuteAction()* message is added to the pattern sequence diagram as an optional message and it is stereotyped $<< optional >>$.

VI. Description of ADAS Action Subsystem Pattern

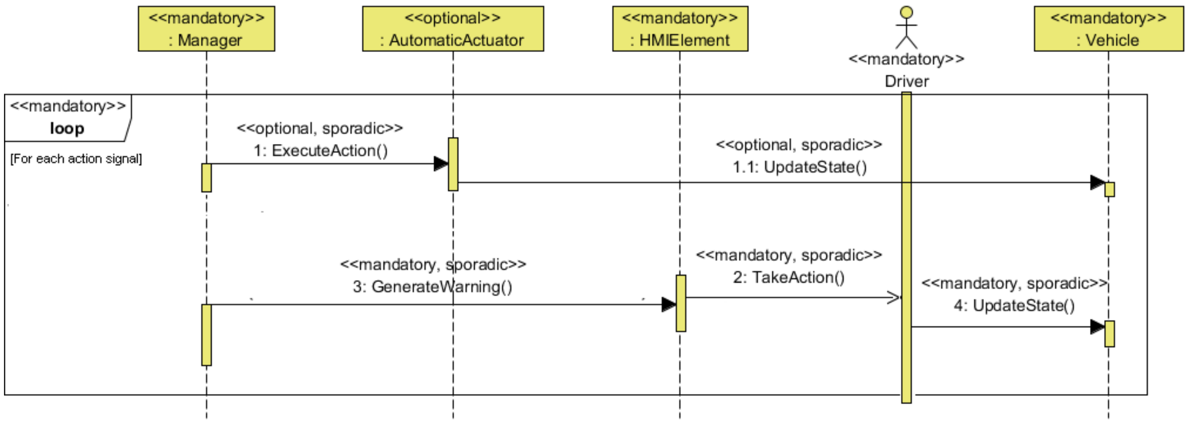
In this section, we describe the proposed pattern through the following elements: name, context, problem, forces and solution.

1) **Name**
ADAS Action Subsystem (ADAS-AS).
2) **Context**
When the system detects a critical situation, it generates warning information to the driver through an HMI and/or it provides automatic actions (e.g., activates the brake actuator or the throttle actuator).
3) **Problem**
How ADAS-AS can be applied to take the actions (automatic actions and/or warning information) that increase the driver's safety and prevent collisions?
4) **Forces**
The action subsystem communicates the warnings to the driver through an appropriate HMI (visual, audible and haptic devices) and/or activates vehicle dynamic actuators (e.g., steering and brakes) according to a potential risk. Driver will be warned early of hazards using different warning levels (e.g., low danger and high danger) to have enough time to take a corrective action. In fact, the warning time depends on the reaction time of each driver.
5) **Solution**
Static specification: Figure 7 presents the action subsystem static view, i.e., the participants represented by the class diagram.
AutomaticActuator. The modeled actuators are devices in a vehicle, used to generate automatic mechanical actions such as brakes actuators which make the vehicle go slow or stop. *AutomaticActuator* class has: (i) *Status* attribute that represents the state's actuator (i.e., an actuator can be activated or deactivated) and (ii) *ReactionTime* attribute that represents the time needed for an actuator to provide automatic actions. This class has an *ExecuteAction()* operation to indicate that the actuator makes an



<<mandatory,extensible>> Manager
<<sporadic>> +Activate()
+Deactivate()
<<optional, extensible>> AutomaticActuator
-Status : String
<<nfp>> -ReactionTime : NFP_Duration
<<sporadic>> +ExecuteAction()
<<mandatory,extensible>> HMIElement
-Location : String
-Status : String
<<nfp>> -RateFA : NFP_Pourcentage
<<sporadic>> +GenerateWarning()
<<mandatory,extensible>> WarningSignalType
-Priority : String
<<nfp>> -Duration : NFP_Duration
-Repetition : Integer
{Partition}
<<optional, extensible>> AudioWarning
-TypeS : String
-Vol : Integer
<<optional, extensible>> HapticWarning
-Freq : NFP_Frequency
-Intensity : Integer
<<optional, extensible>> VisualWarning
-TypV : String
-Luminance : Integer
-Size : Integer
-FlashingRatio : boolean
-Color : String
<<optional, extensible>> AutomaticAction
-Name : String
<<nfp>> -Duration : NFP_Duration
<<mandatory,extensible>> Vehicle
-Description : String
-Status : String
+UpdateState()
<<mandatory,extensible>> WarningSignal
-IdWarning : Integer
<<mandatory,extensible>> DriverAction
-Name : String
<<nfp>> -Duration : NFP_Duration
<<mandatory,extensible>> Driver
<<nfp>> -ReactionTime : NFP_Duration
-Description : String
<<sporadic>> +TakeAction()
<<optional>>
<<mandatory>>

Fig. 7. ADAS-AS pattern.



<<mandatory>> : Manager
<<optional>> : AutomaticActuator
<<mandatory>> : HMIElement
<<mandatory>> Driver
<<mandatory>> : Vehicle
<<mandatory>> loop
[For each action signal]
<<optional, sporadic>> 1: ExecuteAction()
<<optional, sporadic>> 1.1: UpdateState()
<<mandatory, sporadic>> 3: GenerateWarning()
<<mandatory, sporadic>> 2: TakeAction()
<<mandatory, sporadic>> 4: UpdateState()

Fig. 8. Action subsystem sequence diagram.

automatic force to change the vehicle's state, and thus reduces the risk of accidents. This operation is stereotyped $<< \ sporadic \ >>$ to indicate that the action is performed whenever a critical situation is detected. We define an OCL constraint related to the *AutomaticActuator* class. This constraint *(context AutomaticActuator::ExecuteAction() pre: deadline* $\leq$ *current time + D)*, where D is the duration before a risk occurs.

HMIElement. The modeled HMI elements are devices that provide warning information to the driver. They can be a car speaker, a Head-Up Display and a haptic seat. *HMIElement* class has: (i) *Location* attribute that represents the position of the HMI element in the vehicle, (ii) *Status* attribute that represents the state's HMI and (iii) *RateFA* attribute that represents a needless alarm given by a processing error. This Class has a *GenerateWarning()* operation to indicate that an HMI element provides warnings to the driver in order to react. This operation is stereotyped $<< \ sporadic \ >>$ because an HMI element generates warning information only if a danger is detected. We define an OCL constraint related to the *HMIElement* class. This constraint *(context HMIEle-*

ment::GenerateWarning() pre: deadline $\leq$ current time + D + self.Driver.ReactionTime), where D is the duration before a risk occurs. The HMI element generates warnings taking into account the driver's reaction time.

Manager. The manager is responsible for processing the warning provided by the controller. It indicates which HMI hardware components or automatic actuators should be active/inactive.

AutomaticAction. This class represents the different actions provided by an automatic actuator to avoid dangers (e.g., automatic braking).

WarningSignal. The action subsystem provides different warnings if a critical situation is detected. These warnings are generated to the driver through HMI elements.

WarningSignalType. The warning signals are classified into visual, auditory and haptic modes. These types are characterized by (i) a priority that represents the level of the warning according to the degree of hazard (e.g., high warning and low warning). The high priority warning requires an immediate action and should be distinguishable from other warnings, (ii) a duration that constitutes the time interval in which the warning is considered valid and (iii) repetition that represents the repetition rate of the warning.

AudioWarning. Auditory warnings include both acoustic (e.g., tone and auditory icons) and speech outputs. These warnings should be presented in higher frequency.

VisualWarning. Visual outputs can be symbols and/or texts. These warnings should take into account some properties such as luminance, size, flashing rate and color.

HapticWarning. Haptic warnings should be sufficiently intense to make drivers able to feel them. They should be presented in a form that the driver is physically able to perceive them (e.g., steering wheel vibration and accelerator vibration).

Driver. The driver needs to understand the warning signal, to choose an appropriate response and to take action. The driver must react immediately to reduce the risk of accident. This class has *TakeAction()* operation which is stereotyped $<< sporadic >>$ to indicate that the driver take a corrective action only if the system generates warnings.

DriverAction. This class represents the reactions taken by the driver after each warning of a hazard event, such as braking and steering. We use notes to define a constraint under OCL (Object Constraint Language) related to the *DriverAction* class. This constraint *(context DriverAction inv: self.Duration $\leq$ self.WarningSignalType.Duration)* indicates that the driver must react immediately.

Vehicle. This class has the *UpdateState()* operation to indicate that the vehicle changes its status according to any automatic action provided by a mechanical actuator (represented by the association between *Vehicle* class and *AutomaticAction* class) or any action taken by the driver (represented by the association between *Vehicle* class and *DriverAction* class).

Dynamic specification: Figure 8 presents a sequence diagram of the action subsystem pattern. In this diagram, we are interested in modeling the manner to take an action that minimizes the hazards and prevents accidents. In fact, the action subsystem consists of (i) several automatic actuators which provide some automated actions through the *ExecuteAction()* operation such as the brake actuator which activates the brake pedal to decelerate the vehicle, and (ii) different HMI elements which deliver warnings to the driver through the *GenerateWarning()* operation. The driver takes the appropriate decision according to the generated warning through the *TakeAction()* operation. For example, if a system detects a risk of frontal collision, it provides warnings indicating that the driver must decelerate to avoid the collision. When the action (automatic action or warning) is taken, the controlled vehicle updates its state through the *UpdateState()* operation. *ExecuteAction()*, *GenerateWarning()*, *TakeAction()* and *UpdateState()* operations are stereotyped $<< sporadic >>$ to indicate that the actions are triggered only if the system detects an hazard.

VII. Conclusion and future work

The main objective of this work was to define a RT design pattern specific to the action subsystem of ADAS. This pattern, named ADAS-AS, models the structural, behavioral and real time aspects of an action subsystem. In fact, it models the different components of the actuator subsystem of ADAS such as the automatic actuators and the HMI elements. This pattern facilitates the modeling of any ADAS; it will be easy for the designer to reuse this pattern by adapting it to the needs of a particular advanced driver assistance system without starting from scratch. Therefore, modeling using ADAS-AS reduces the system failure. However, using another pattern such as [6][10][16] allows the designer to add all specificities of an ADAS. So, the developed system does not meet the requirements.

In future work, we will propose to reuse the following patterns to model real industrial ADAS systems: the data acquisition pattern proposed in [9], the controller pattern defined in [10] and the ADAS-AS pattern defined in this paper. We will propose also to develop a tool that supports the definition of ADAS patterns and the dictionary of the semantic relations in the design process. Then, we will propose an approach of using patterns to build a new architecture of an ADAS system that integrates a RT database. This approach helps designers to build their systems without starting from scratch. The developer is limited to provide the properties and the specificities related to a particular system.

References

[1] W. Prestl, T. Sauer, J. Steinle, and O. Tschernoster, "The BMW active cruise control ACC," SAE transactions, vol. 109, no. 7, 2000, pp. 119-125, doi:10.4271/2000-01-0344.

[2] E. Johansson, E. Karlsson, C. Larsson, and L. Eriksson, "Implementation and evaluation of lane departure warning and assistance systems," Advances in Human Aspects of Road and Rail Transportation, Edited by Neville A . Stanton, 2012, pp. 37-46.

[3] M. R. Ghahroudi and R. Sabzevari, ”Sensor data and fusion,” Vienna, Austria: I-Tech Education and Publishing KG, ch. Multisensor Data Fusion Strategies for Advanced Driver Assistance Systems, 2009, pp. 141-166.

[4] F. Biral, M. D. Lio, R. Lot, and R. Sartori, ”An intelligent curve warning system for powered two wheel vehicles,” European Transport Research Review, vol. 2, no. 3, Dec. 2010, pp. 147-156.

[5] L. Danielsson, H. Lind, E. Bekiaris, M. Gemou, A. Amditis, M. Miglietta, and P. Stålberg, ”HMI principles for lateral safe applications,” in Universal Access in Human-Computer Interaction, ser. Lecture Notes in Computer Science, C. Stephanidis, Ed. Springer-Verlag Berlin, Heidelberg, vol. 4555, July. 2007, pp. 330-338.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, ”Design patterns: Elements of Reusable Object-Oriented Software,” Addison-Wesley Edition, 1994.

[7] A. Armoush, F. Salewski, and S. Kowalewski, ”Design pattern representation for safety-critical embedded systems,” Journal of Software Engineering and Applications, vol. 2, April. 2009, pp. 1-12.

[8] P. D. Bruce, ”Real-Time Design Patterns: Robust Scalable Architecture for Real-Time Systems”, Addison-Wesley Edition, 2002.

[9] H. Marouane, A. Makni, R. Bouaziz, C. Duvallet, and B. Sadeg, ”A real-time design pattern for advanced driver assistance systems,” in Proceedings of 17^{th} European conference on Pattern Languages of Programs (EuroPLoP 2012), 2012, pp. C6:1-C6:11.

[10] S. Rekhis, N. Bouassida, C. Duvallet, R. Bouaziz, and B. Sadeg, ”A UML-profile for domain specific patterns: Application to real-time,” in DE@CAISE’10: the Domain Engineering workshop of the 22^{nd} International Conference on Advanced Information Systems Engineering (CAiSE’10), Hammamet, Tunisia, 2010, pp. 32-46

[11] S. Rekhis, N. Bouassida, C. Duvallet, R. Bouaziz, and B. Sadeg, ”A Process to Derive Domain-Specific Patterns: Application to the Real-Time Domain,” Proceedings of 14^{th} International Conference on Advances in Databases and Information Systems (ADBIS’2010), Sept. 2010, pp. 475-489.

[12] S. Rekhis, N. Bouassida, C. Duvallet, R. Bouaziz, and B. Sadeg, ”Modeling real-time applications with reusable design patterns,” International Journal of Advanced Science and Technology (IJAST), vol. 22, 2010, pp. 71-86.

[13] D. C. Schmidt, M. Stal, H. Rohnert, and F. Buschmann, ”Pattern-Oriented Software Architecture: Patterns for Concurrent and Networked Objects,” 2^nd ed. New York, NY, USA: John Wiley Sons, Inc., 2000.

[14] D. Slutej, J. Hakansson, J. Suryadevara, C. Seceleanu, and P. Pettersson, ”Analyzing a pattern-based model of a real-time turntable system,” in 6^{th} International Workshop on Formal Engineering approaches to Software Components and Architectures(FESCA), ETAPS’09, York, UK, B. Z. Jens Happe, Ed., Electronic Notes in Theoretical Computer Science (ENTCS), Elsevier, vol. 253, Oct. 2009, pp. 161-178.

[15] K. Soundararajan and R. W. Brennan, ”Design patterns for real-time distributed control system benchmarking,” Journal of Robotics and Computer-Integrated Manufacturing, vol. 24, no. 5, Oct. 2008, pp. 606-615.

[16] S. Konrad and B. H.C. Cheng, ”Requirements Patterns for Embedded Systems,” Proceedings of the IEEE Joint International Conference on Requirements Engineering (RE02), Sept. 2002, pp. 127-136.

[17] OMG, ”Unified modeling language (UML) infrastructure,” v2.1.2, formal/2007-11-04, 2007.

[18] M. Clauß and I. Jena, ”Modeling variability with UML,” in In GCSE 2001 Young Researchers Workshop, 2001.

[19] N. Idoudi, C. Duvallet, R. Bouaziz, B. Sadeg, and F. Gargouri, ”Structural model for real-time databases: an illustration,” in Proceedings of 11^{th} IEEE International Symposium on Object-oriented Real-time distributed Computing (IEEE ISORC’2008). Orlando, United States: IEEE Computer Society Washington, May. 2008, pp. 58-65.

[20] OMG, ”A UML profile for MARTE,” 2007.

[21] OMG, ”UML 2.0 OCL specification,” 2007.

[22] K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, ”Software reuse metrics for object-oriented systems,” in Proceedings of the third ACIS International Conference on Software Engineering Research, Management and Applications (SERA’05). IEEE computer society, Aug. 2005, pp. 48-54.

[23] N. Bouassida, H. Ben-Abdallah, F. Gargouri, and A. Ben-Hamadou, ”A stepwise framework design process,” in IEEE International Conference on Systems Man and Cybernetics, Hammamet, Tunisia, 2002.

[24] E. D. Bekiaris, A. Spadoni, and S. I. Nikolaou, ”Saferider project: new safety and comfort in powered two wheelers,” in Proceedings of the 2^{nd} conference on Human System Interactions. Piscataway, NJ, USA: IEE press, May. 2009, pp. 600-602, doi: 10.1109/HSI.2009.5091045.

[25] J. P. Diederichs, M. Fontana, G. Bencini, S. Nikolaou, R. Montanari, A. Spadoni, H. Widlroither, and N. Baldanzini, ”New HMI concept for motorcycles - the saferider approach,” in Proceedings of the 8^{th} International Conference on Engineering Psychology and Cognitive Ergonomics: Held as Part of HCI International 2009. Springer-Verlag Berlin, Heidelberg, July 2009, pp. 358-366.

Metrics for Measuring Quality of Real-time Design Patterns

Saoussen Rekhis*, Hela Marouane*, Rafik Bouaziz†, Claude Duvallet* and Bruno Sadeg*
*University of Le Havre
Le Havre, France
Email: {Hela.Marouane, Claude.Duvallet, Bruno.Sadeg}@litislab.fr
†University of Sfax
Sfax, Tunisia
Email: {Saoussen.Rekhis, Raf.Bouaziz}@fsegs.rnu.tn

***Abstract*—In recent years, the influence of domain specific design patterns on software quality has attracted increasing attention in the area of software engineering. Indeed, such patterns facilitate the development process of systems, leading to efficient solutions for a particular domain. Since the usage of such patterns has been recommended, there is a need to evaluate their efficiency in a domain, i.e., they answer the question if the provided model encapsulates really the concepts tied to a particular domain. It is also important to determine the amount of pattern elements reuse in order to verify that the patterns cover the majority of domain concepts. The amount of reuse metrics determine how much pattern elements are reused in a designed system, whereas reusability metrics are intended to measure the degree to come up with the specificities of a particular domain in the patterns. Our proposal aims to adapt some existing reuse metrics and to define new metrics for reusability assessment. The usage of these metrics is illustrated through a case study in real-time domain.**

Keywords*—Reusability metrics; Amount of reuse metrics; Design pattern.

I. INTRODUCTION

Reusability and software reuse are two major aspects in object oriented software. Reusability is the possibility that an artifact can be reused, i.e., the fitness of an artifact to be reusable. Software reuse is the use of existing software components to build new systems, rather than designing and implementing from scratch. It provides significant improvements in software productivity and quality during the life cycle of a system. Software reuse is supported by different approaches including frameworks, product lines, patterns and program libraries. Our research focuses on the reuse of design patterns that are applied in a specific domain (e.g., real-time domain). These patterns offer flexible architectures with clear boundaries, in terms of well-defined and highly encapsulated parts that are in alignment with the natural constraints of the domain [1]. They present a successful mechanism to capture and promote best practices in the software design. These reasons motivated several researchers on the definition and the application of domain specific design patterns [1][2][3]. However, **these researchers do not provide** a quantitative evaluation of effectiveness of applying these patterns.

In this paper, we have seen necessary (i) to adapt two existing metrics namely, Class Template Factor (CTF) and **Function Template Factor (FTF) [4] to measure, respectively,** the amount of pattern classes reuse and operations reuse in a given application and (ii) to add a new metric to compute the amount of attributes reuse since each class in a pattern has two essential parts, corresponding to its attributes and its operations. Moreover, we are interested in defining new metrics for assessing reusability of domain specific design patterns. The aim of these metrics is to determine whether the patterns represent the concepts that suit a specific domain; they compute the degree to meet the concepts related to this domain. These metrics are applied in a validation step of domain specific design patterns. After their definition, these patterns are used to model different systems in the considered domain. For each pattern reuse, we compute the amount of reuse metrics and reusability metrics in order to evaluate the quality of patterns. Indeed, it is essential to show that the application designers need only to add some system specific elements since the majority of application elements are reused from the patterns. Without computing the amount of reuse metrics, we are not sure that the patterns cover the majority of domain concepts. Thereafter, we consider an example of a real-time application (the freeway traffic control application) designed without and with using a domain specific pattern (sensor pattern [5]), as the base for the illustration of the defined metrics. We also interpret how the values measured of these metrics may contribute and be used effectively to evaluate the quality of the sensor pattern.

The remainder of this paper is organized as follows. Section II presents related work. The definition of metrics for the measure of the amount of domain specific design patterns reuse and the reusability assessment is described in Section III. The explanation of these metrics is presented in Section IV. This latter gives a case study to illustrate these metrics. The evaluation of applying the sensor pattern is described in Section V. Finally, Section VI concludes the paper and outlines our future work.

II. RELATED WORK

The use of reusable components (e.g., design patterns) provides a key element in improving the way software is developed and supported over its life cycle. Design and software reuse reduce development efforts and increase the quality of developed systems. In this context, a critical issue is to identify and qualify reusable components. For these reasons, several works on reuse and reusability metrics have been proposed.

A. Amount of reuse metrics

Frakes et *al.* postulated in [6] that ”amount of reuse metrics are used to assessing and also monitoring the reuse improvement effort by tracking of the percentages of reuse for life cycle objects". These metrics aim to determine how much reuse is present within a given system. The common form of these metrics is defined as the ratio between the amount of the life cycle object reused and the total size of the life cycle object [6]. However, there are many ways to implement this metric. Each way provides different aspects of the reuse ranging from how much code is reused to how often it is reused. For example, the amount of code reuse is defined as the ratio between the number of reused lines of code in a system and the total lines of code in a system.

Frakes *et al.* have shown in [7] various implementations of reuse level (RL) and reuse frequency (RF) metrics which have been proposed in [6] for measuring amount of reuse. RL and RF metrics are measured relative to different granularity of items (e.g., line of codes, functions, files and projects) of source software (i.e., C, java and C++).

Zaigham *et al.* [8] analyze the existing amount of reuse metrics on the basis of their industrial applicability. These metrics are applied to different software projects written in C++ to provide a complete understanding of the level of correlation that exists between them and other software metrics such as cyclomatic complexity, volume and lines of code.

Aggarwal *et al.* have proposed in [4] two metrics for measuring amount of reuse in object oriented software using generic programming in the form of templates. The first metric, called CTF, is defined as a ratio between the number of classes using class templates and the total number of classes in a source code. The second metric, called FTF, is defined as a ratio between the number of functions using function templates and the total number of functions.

These works are focused on different reuse metrics, aiming to measure the amount of reuse of software components and to determine the portion of the new or modified code and the portion of the reused code. These metrics only deal with source code which is typically available at the later stages of the software life cycle, failing to address the importance of the software *artifacts* produced during earlier stages such as analysis and design. So, we see that it is necessary to define metrics for the assessment of the level of design structures reuse in application models. In fact, analysis and design are crucial phases in software development, because they heavily influence the cost of the implementation and maintenance phases. Thus, we intend hereafter to adapt existing reuse metrics defined in [4] for measuring the amount of patterns reuse in applications designed with UML. Moreover, we will add another metric to compute the amount of attributes reuse.

B. Reusability assessment

Reusability metrics indicate the possibility that a component is reusable and enable to identify a good quality of a component for reuse, but, they don't provide a measurement of how many components are reused.

Different studies are based on the definition of reusability metrics.

Bhatia *et al.* [9] have proposed an approach to measure the reusability of a class diagram based on Depth of Inheritance Tree (DIT) of a class, Number of Children (NOC) and Coupling Between Object classes (CBO) metrics [10]. This approach consists to define a formula for reusability based on the principle that DIT and NOC have positive effect on reusability, whereas CBO has negative impact on reusability of a class. The authors consider that reusability of a class diagram is equal to the maximum reusability of a class in the diagram.

Gill *et al.* [11] have proposed new metrics which can be computed from inheritance hierarchies: Breadth of Inheritance Tree (BIT), Method Reuse Per Inheritance Relation (MRPIR), Attribute Reuse Per Inheritance Relation (ARPIR), Generality of Class (GC) and Reuse Probability (RP). BIT metric is compared to two existing metrics (DIT [10] and NOC [10]) to indicate that this metric measures the breadth of the whole inheritance tree, not to compute the number of immediate sub classes of a class. MRPIR and ARPIR metrics are compared respectively to Method Inheritance Factor (MIF) [12] and Attribute inheritance Factor (AIF) [12] to highlight that these two proposed metrics give clearer picture of reuse due to inheritance. In fact, MRPIR metric (respectively ARPIR metric) computes average number of reused methods (respectively attributes) in inheritance hierarchy and not in all classes. GC metric considers the generality of the class as feature of reusability whereas DIT does not consider characteristics of the class.

Subedha *et al.* [13] have used reuse utility percent and reuse frequency metrics as the assessment attributes for reusability of the software component in context level. These metrics determine which components have high reuse potential from a set of standard components in an existing environment.

The previous metrics estimate the probability of reusability of a component and evaluate its design quality (e.g., when CBO increases, reusability decreases and it becomes harder to modify the software system). These metrics indicate that whether or not the components are reusable in the future. **But, they do not answer an essential question:** Do the reusable components represent the specificities of a particular domain? In order to fill this lack, we propose in this paper other metrics for reusability assessment of domain specific design patterns. The aim of these metrics is to show if these patterns are well-defined and they take into account the concepts of the considered domain.

III. METRICS DEFINITION

In this section, we adapt some existing metrics related to the amount of reuse for class diagrams. We also define new metrics that determine the reusability of patterns, i.e., the probability of their reuse.

A. Amount of reuse metrics

We have to adapt the pair of metrics CTF and FTF [4] to compute how much classes and operations to reuse are present within a given application. Moreover, we consider as important to add a new metric, called Attribute Reuse Level, to measure reuse level of attributes in each class of a system. In fact, attributes are essential elements that represent the properties of a class.

The values of these metrics range from 0 to 1. When reuse of patterns elements increases, the reuse level value approaches to 1. A reuse level of 0 indicates no reuse of pattern elements.

1) Metric 1: Class Reuse Level (CRL): This metric is defined as the ratio between the number of reused pattern classes (*RPC*) and the total number of classes in the designed system as shown in (1).

Let us consider a model, with *n* classes C_1, C_2, ..., C_n.

$$CRL = \frac{\sum_{i=1}^{n} RPC(C_i)}{n} \qquad (1)$$

where,

$$RPC(C_i) = \begin{cases} 1 & \text{if } the\ class\ is\ reused\ from\ a\ pattern, \\ 0 & \text{otherwise.} \end{cases}$$

2) Metric 2: Attribute Reuse Level (ARL): This metric is defined as the ratio between the number of reused attributes (*RAT*) of pattern classes and the total number of attributes in the designed system as shown in (2).

Let us consider a model having *n* classes C_1, C_2, ..., C_n and m_i attributes a_1, a_2, ..., a_{m_i} for each class C_i.

$$ARL = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m_i} RAT(a_{ij})}{\sum_{i=1}^{n} m_i} \qquad (2)$$

where,

$$RAT(a_{ij}) = \begin{cases} 1 & \text{if } the\ attribute\ is\ reused \\ & \quad from\ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

3) Metric 3: Operation Reuse Level (ORL): This metric is defined as the ratio between the number of reused operations (*ROP*) of pattern classes and the total number of operations in **the designed system, as shown in (3)**.

Let us consider a model having *n* classes C_1, C_2, ..., C_n and k_i operations op_1, op_2, ..., op_{k_i} for each class C_i.

$$ORL = \frac{\sum_{i=1}^{n} \sum_{q=1}^{k_i} ROP(op_{iq})}{\sum_{i=1}^{n} k_i} \qquad (3)$$

where,

$$ROP(op_{iq}) = \begin{cases} 1 & \text{if } the\ operation\ is\ reused \\ & \quad from\ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

B. Reusability Metrics

We propose new metrics which indicate the possibility that a pattern can be reused in new systems. Moreover, these metrics indicate whether these patterns allow designing the specificities tied to this domain or not. They are calculated from two releases of each application. Release 1 is designed without using any pattern. Release 2 is designed using design patterns. Measurement values of these metrics are always normalized to a number between 0 and 1. When metric values approach to 1, this means that the majority of the pattern elements (i.e., classes, attributes and operations) are recognized in the systems which are designed without using patterns. Thus, the patterns support the requirements related to a particular domain. Otherwise, the value 0 indicates that no pattern elements are identified in the systems designed without using patterns.

1) Metric 1: Class Reusability (CR): The metric CR is defined as the ratio between the number of identified pattern classes (*IPC*) in a model designed without using patterns and the number of reused pattern classes (*RPC*) in this model when designed using patterns as shown in (4).

Let us consider a model with *n* classes C_1, C_2, ..., C_n.

$$CR = \frac{\sum_{i=1}^{n} IPC(C_i)}{\sum_{i=1}^{n} RPC(C_i)} \qquad (4)$$

where,

$$IPC(C_i) = \begin{cases} 1 & \text{if } the\ class\ is\ identified\ as \\ & \quad a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

$$RPC(C_i) = \begin{cases} 1 & \text{if } the\ class\ is\ reused\ from\ a\ pattern, \\ 0 & \text{otherwise.} \end{cases}$$

2) Metric 2: Attribute Reusability (AR): The metric AR is defined as the ratio between the number of identified attributes (*IAT*) of pattern classes in a model designed without using patterns and the number of reused attributes (*RAT*) of pattern classes in this model when designed using patterns as shown (5).

Let us consider a model with *n* classes C_1, C_2, ..., C_n and m_i attributes a_1, a_2, ..., a_{m_i} for each class C_i.

$$AR = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m_i} IAT(a_{ij})}{\sum_{i=1}^{n} \sum_{j=1}^{m_i} RAT(a_{ij})} \qquad (5)$$

where,

$$IAT(a_{ij}) = \begin{cases} 1 & \text{if } the\ attribute\ is\ identified\ as \\ & \quad an\ attribute\ of\ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

$$RAT(a_{ij}) = \begin{cases} 1 & \text{if } the\ attribute\ is\ reused\ from \\ & \quad a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

3) Metric 3: Operation Reusability (OR): The metric OR is defined as the ratio between the number of identified operations (*IOP*) of pattern classes in a model designed without using patterns and the number of reused operations (*ROP*) of pattern classes in this model when designed using patterns as shown in (6).

Let us consider a model with *n* classes C_1, C_2, ..., C_n and k_i operations op_1, op_2, ..., op_{k_i} for each class C_i.

$$OR = \frac{\sum_{i=1}^{n}\sum_{q=1}^{k_i} IOP(op_{iq})}{\sum_{i=1}^{n}\sum_{q=1}^{k_i} ROP(op_{iq})} \qquad (6)$$

where,

$$IOP(op_{iq}) = \begin{cases} 1 & \text{if } the\ operation\ is\ identified\ as \\ & \quad an\ operation\ of\ a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

$$ROP(op_{iq}) = \begin{cases} 1 & \text{if } the\ operation\ is\ reused\ from \\ & \quad a\ pattern\ class, \\ 0 & \text{otherwise.} \end{cases}$$

IV. CASE STUDY

In this section, we present a case study as an example to explain the application of reusability and reuse metrics. The measurement of these metrics was carried out in a pattern specific to the real-time domain [5] **(Figure 1)**. We consider this domain as the base of the illustration of the defined metrics since the design of real-time systems is considered to be a complex process, as all components and real-time constraints have to be considered during the design phase. In fact, real-time applications must be able to meet real-time constraints, i.e., they have to guarantee that each action meets its deadline and that data are used during their validity interval. Thus, it is necessary to give a great importance to real-time applications design.

The real-time domain consists of three functionalities: (1) acquisition of data from environment, (2) data analysis and control and (3) sending orders and commands to actuators. For each functionality, we have defined a design pattern that captures RT domain knowledge and design expertise. In this paper, we present only the sensor pattern [5], which focuses on the modeling of data acquisition functionality of real-time domain. This pattern is applied to model the freeway traffic control application.

A. Application description

The COMPASS [3] is a freeway traffic management system intended to improve safety and to provide a better level of service to motorists. According to this system, the current traffic state is obtained from sensors installed in the freeway: inductance loop detectors and supervision cameras. In fact, inductance loop detectors are embedded in the pavement. Their shape may vary depending on the system requirements. Inductance loop detectors, which are active sensors, measure speeds and lengths of vehicles, number of vehicles and occupancy rates of road segments. These acquired measures are updated and transmitted periodically to the Central Computer System to monitor traffic and to identify traffic incidents. Whereas the Closed Circuit Television (CCTV) supervision cameras constitute passive sensors that transmit periodically the images to the Traffic Operation Centre (TOC). These images are used to confirm the reception of data through the inductance loop detectors and to provide information on local conditions. CCTV cameras are normally mounted on the top of 15 meters poles at approximately 1 km apart along the freeway. These cameras are characterized by a resolution 126 x 185 pixels. Each measure, taken from the environment of this system, has a value, a timestamp and a validity interval to verify the temporal consistency of the collected traffic data. In addition, the minimum and maximum thresholds of each taken measure must be defined in order to determine the abnormal values for which COMPASS system may detect an incident.

The data acquisition subsystem of this application is designed without and with using sensor pattern to calculate the reusability metrics defined in Subsection III-B. Whereas the amount of reuse metrics **(Subsection III-A)** are computed based on this application model reusing the pattern. Figures 2 and 3 show respectively the model without and with the use of the pattern. The model without using pattern is designed by three professors who have an experience in UML.

B. Metrics illustration

Table I shows all metrics calculated from the models of freeway traffic management application already presented (c.f. Figures 2 and 3).

TABLE I
REUSE AND REUSABILITY METRICS CALCULATIONS.

	Metrics	Value
Amount of reuse metrics	Class Reuse Level (CRL)	$\frac{5}{9} = 0.55$
	Attribute Reuse Level (ARL)	$\frac{10}{14} = 0.71$
	Operation Reuse Level (ORL)	$\frac{6}{9} = 0.67$
Reusability metrics	Class Reusability (CR)	$\frac{5}{5} = 1$
	Attribute Reusability (AR)	$\frac{7}{10} = 0.7$
	Operation Reusability (OR)	$\frac{4}{6} = 0.67$

According to the model presented in Figure 2, we identify the following classes as elements of sensor pattern: (i) *Sensor*, *InductanceLoop* and *Camera* classes: **they play, respectively,** the role of *Sensor*, *Active_Sensor* and *Passive_Sensor* classes, (ii) *RoadSegment* and *Vehicle* classes: they correspond to the *ObservedElement* class and (iii) *trafficData* class: it matches the *Measure* class. As the *RoadSegment* and *Vehicle* classes play the same role (i.e., *ObservedElement* class), they are

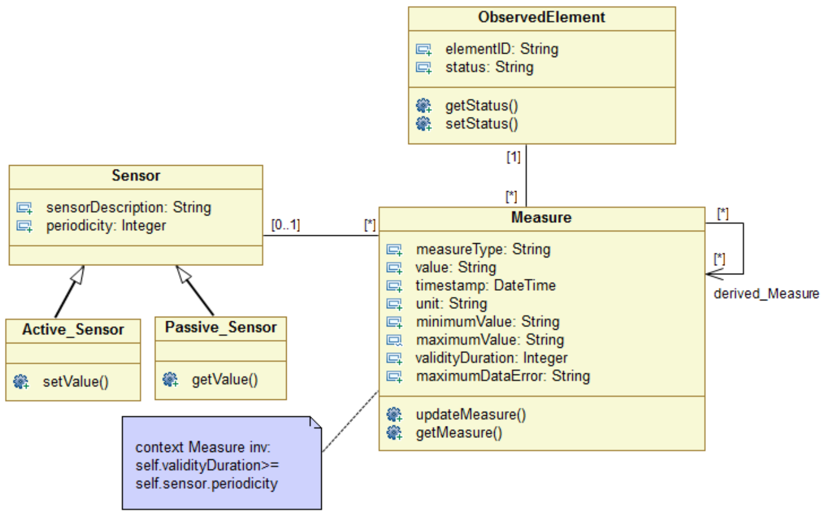


Fig. 1. Sensor pattern [5].

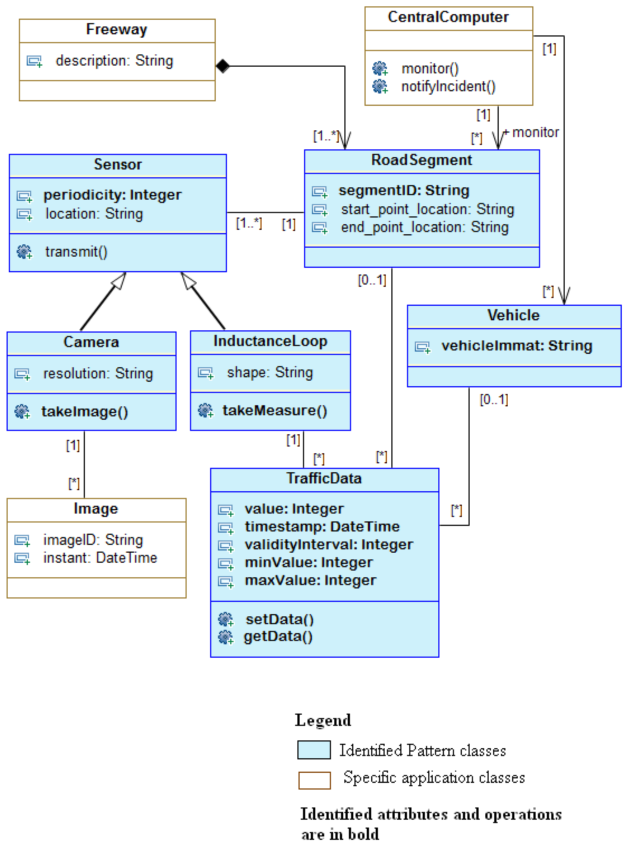


Fig. 2. Data acquisition subsystem of COMPASS without using sensor pattern.

calculated as only one identified pattern element. In other words, the number of identified (*RoadSegment*,*Vehicle*) is equal to 1 (not to 2). This avoids any conflicts in the evaluation of conceptual models with various correct solutions. Indeed, if the elements have the same role in a system, the designer has two solutions: he may use or not the inheritance relationships. Thus, we must calculate the number of identified elements considering the classes which play the same role as one element to obtain the same measurement.

Sensor class has one attribute corresponding to the attribute of the Sensor pattern class: it is *periodicity*. The *takeImage()* operation of *Camera* class matches the *getValue()* operation of *Passive_Sensor* pattern class. Whereas, the *takeMeasure()* operation of *InductanceLoop* class correspond to the *setValue()* operation of *Active_Sensor* pattern class. *RoadSegment* and *Vehicle* **classes have, respectively,** *segmentId* and *vehicleImmat* attributes that correspond to *elementId* attribute of *ObservedElement* pattern class. All attributes and operations

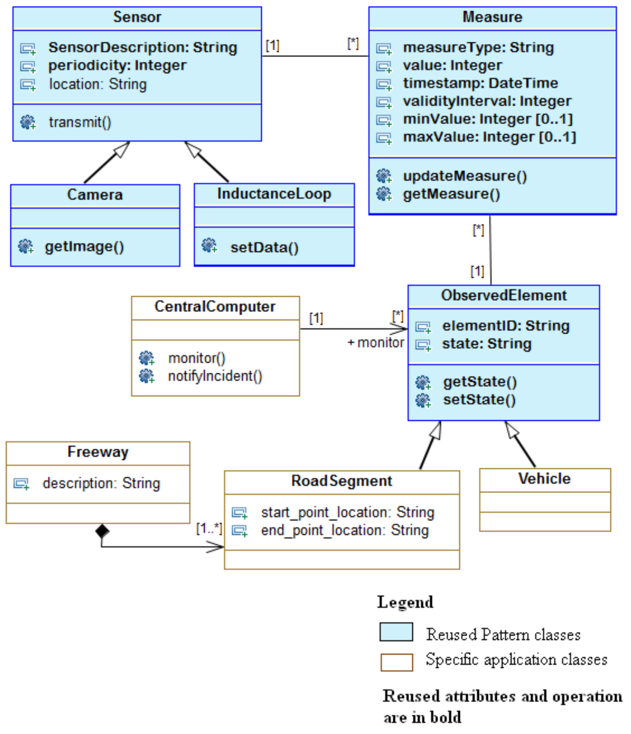


Fig. 3. Data acquisition subsystem of COMPASS with using sensor pattern.

of *TrafficData* class are elements of *Measure* pattern class. The identification of model elements (classes, attributes and operations) as pattern participants is based on a semantic comparison between classes names using a domain dictionary. This dictionary holds for each term (i.e., a class name, an attribute name and operation name) the possible synonyms, antonyms, hypernyms. The construction of this dictionary requires the intervention of pattern designers to determine the linguistic relations for each introduced pair of terms. The designer specifies, for example, that the class name observedElement is the hypernym of vehicle class name.

The reusability metric values presented in Table I approach to 1, it indicates that the majority of pattern participants are recognized in the application model. If we obtain the same results in several case studies, this means that the patterns cover the domain concepts.

As shown in Figure 3, the image is considered as a measure taken by a camera sensor. It has a value (the taken photo), a timestamp and a validity duration. But, it does not have minimum and maximum values. Thus, *minVal* and *maxVal* attributes have the multiplicity [0..1]. The other attributes have the default multiplicity [1].

We have reused the classes *Sensor*, *Active_Sensor*, *Passive_Sensor*, *ObservedElement*, and *Measure*. *RoadSegment* and *Vehicle* classes constitute specific application elements which specialize *ObservedElement* class. This class reuses all features of *ObservedElement* pattern class. We have also reused all attributes and operations of *Measure* class except *Maximum Data Error* attribute. From *Sensor* class, we have instantiated *description* and *periodicity* attributes. In addition, the reused operations of *Active_Sensor* and *Passive_Sensor* classes correspond respectively to *setData()* and *getImage()* operations of *InductanceLoop* and *Camera* classes.

The reuse metric values presented in Table I mean that the majority of application elements (classes, attributes and operations) are reused from the pattern and a limited number of application specific elements are added. This result is approved in the next Section by applying the sensor pattern in several real-time applications.

V. Sensor Design Pattern Evaluation

We present in Table II the values of reuse metrics and reusability metrics obtained for the sensor pattern which is used for modeling ten different real-time applications that we reference A1, A2, ... , A10. On one hand, the values obtained for reuse metrics show that more than half of the classes, the attributes and the operations of real-time applications corresponding to the sensor pattern are instantiated from this pattern. For example, the values of reuse metrics obtained in Table II for the application A1 show that 83% of classes (CRL = 0,83), 62% of attributes (ARL = 0,62) and 87% operations (ORL = 0,87) belonging to the model fragment relative to the sensor pattern are instantiated from this pattern. There are even cases (applications A7, A8 and A9) where all applications classes are instances of pattern classes (CRL = 1). Thus, we deduce a good level of reuse of the sensor pattern elements in the modeling of real-time applications.

On the other hand, the values obtained for reusability metrics calculated for the sensor pattern indicate that the degree of reusability of classes and attributes is better than the reusability of operations. Indeed, we identified all the classes of the sensor pattern (CR = 1) in seven cases of real-time applications modeled without reusing this pattern. In addition, we have identified the majority of the attributes reused from the

TABLE II
RESULTS FOR REUSE METRICS AND REUSABILITY METRICS CALCULATED FOR SENSOR PATTERN.

		A1	**A2**	**A3**	**A4**	**A5**	**A6**	**A7**	**A8**	**A9**	**A10**	**Averge**
Amount of reuse metrics	CRL	0,83	0,57	0,7	0,71	0,8	0,83	1	1	1	0,71	0,81
	ARL	0,62	0,59	0,72	0,6	0,83	0,75	0,75	0,76	0,87	0,73	0,72
	ORL	0,87	0,71	0,83	0,75	0,83	0,7	0,83	0,83	0,72	1	0,80
Reusability metrics	CR	1	1	1	1	1	0,6	1	1	0,8	0,8	0,92
	AR	0,8	0,76	0,66	0,73	0,7	0,76	0,77	0,8	0,64	0,81	0,74
	OR	0,71	0,60	0,5	0,66	0,4	0,42	0,6	0,4	0,37	0,5	0,51

pattern classes. For example, the values of reusability metrics obtained in Table II for application A1 show that all reused classes of the sensor pattern are identified (CR = 1), 80% of the attributes are identified (AR = 0,8) and 71% of operations are identified (OR = 0,71). This means that the reuse of this pattern is interesting in the real-time domain because it adequately represents the concepts of data acquisition functionality.

VI. CONCLUSION AND FUTURE WORK

The main objective of this work is to define two categories of metrics that are important for reuse design. The first one aims to assess the reuse level of pattern participants. When the measurement of amount of reuse metrics increases, it means that the pattern elements are simply reused in the model with a minimal possibility for modification. The second category focuses on predicting the reusability of domain specific design patterns. This kind of metrics checks the presence of pattern elements in a system designed without the usage of patterns. When the measurement of reusability metrics increases, it means that the patterns well represent the domain concepts. Reuse and reusability metrics are then illustrated using a case study and they are calculated for ten applications to evaluate the quality of the sensor pattern. The values of reuse metrics show a high degree of the pattern elements reuse (i.e., more than 70% of classes, attributes and operations of the considered applications are modeled by reusing the sensor pattern in the majority of cases). Similarly, the values obtained for reusability metrics show that the attributes, the operations and especially the classes of the sensor pattern are identified in the applications models designed without reuse of this pattern. Thus, we can conclude that it has a good ability to be reused for modeling real-time applications.

In future work, we will check and evaluate the effectiveness of applying other domain specific design patterns (controller and actuators patterns) for real-time systems based on the measurement of these metrics taken for different case studies.

REFERENCES

[1] D. Port, ”Derivation of domain specific design patterns,” USC Center for software engineering, 1998.

[2] H. Marouane, A. Makni, R. Bouaziz, C. Duvallet, and B. Sadeg, ”A real-time design pattern for advanced driver assistance systems,” in 17^{th} European conference on Pattern Languages of Programs (EuroPLoP), 2012, pp. C6:1-C6:11.

[3] S. Rekhis, N. Bouassida, C. Duvallet, R. Bouaziz, and B. Sadeg, ”A process to derive domain-specific patterns: Application to the real-time domain,” in Proceedings of 14^{th} International Conference on Advances in Databases and Information Systems (ADBIS), 2010, pp. 475-489.

[4] K. K. Aggarwal, Y. Singh, A. Kaur, and R. Malhotra, ”Software reuse metrics for object-oriented systems,” in Proceedings of the third ACIS International Conference on Software Engineering Research, Management and Applications (SERA'05). IEEE computer society,2005, pp. 48-54.

[5] S. Rekhis, N. Bouassida, C. Duvallet, R. Bouaziz, and B. Sadeg, ”Modeling real-time applications with reusable design patterns,” International Journal of Advanced Science and Technology (IJAST), vol. 22, 2010, pp. 71-86.

[6] W. Frakes and C. Terry, ”Software reuse: metrics and models,” ACM Comput. Surv., vol. 28, no. 2, Jun. 1996, pp. 415-435.

[7] W. B. Frakes, R. Anguswamy, and S. Sarpotdar, ”Reuse ratio metrics RL and RF,” in 11^{th} International Conference on Software Reuse, Falls Church, VA, USA, 2009.

[8] M. Zaigham and R. Tauseef, ”Correlation between amount-of-reuse metrics and other software measures with respect to programming code in c++,” Software Quality Control, vol. 11, no. 4, **Nov. 2003**, pp. 301-312.

[9] K. B. Pradeep and M. Rajbeer, ”An approach to measure software reusability of OO design,” in Proceedings of $2^{n}d$ National Conference on Challenges & Opportunities in Information Technology (COIT-2008) RIMT-IET, Mandi GobindgarhA, 2008.

[10] S. R. Chidamber and C. F. Kemerer, ”A metrics suite for object oriented design,” IEEE Transaction on Software Engineering, vol. 20, no. 6, Jun. 1994, pp. 476-493.

[11] S. G. Nasib and S. Sunil, ”Inheritance hierarchy based reuse & reusability metrics in oosd,” International Journal on Computer Science and Engineering (IJCSE), vol. 3, no. 6, 2011, pp. 2300-2309.

[12] R. Harrison, S. Counsell, and R. Nithi, ”An evaluation of the MOOD set of object oriented software metrics,” IEEE Transaction on Software Engineering, vol. 24, no. 6, **1998**, pp. 491-496.

[13] V. Subedha and S. Sridhar, ”Design of a conceptual reference framework for reusable software components based on context level,” International Journal of Computer Science Issues (IJCSI), vol. 9, no. 3, 2012, pp. 26-31.

Using a New UML Profile for Modeling Software Tests

Andrew Diniz da Costa, Carlos José Pereira de Lucena, Ricardo Venieris, Gustavo Carvalho
Laboratory of Software Engineering
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Brazil
{acosta, lucena}@inf.puc-rio.br, {rvenieris, guga}@les.inf.puc-rio.br

***Abstract*—The development of complex systems is becoming extremely common; hence, is motivating the work on software testing. When a large number of tests must be executed to validate the release of a system, several data should be used to correctly coordinate the execution of these tests, such as knowing (i) if the current version of a particular test has been updated, (ii) the interdependence between tests, (iii) the order of execution to be followed, (iv) the priority, (v) the risks associated with the tests, etc. Based on this concern for providing and documenting useful data for the coordination of test execution, this paper offers a new modeling language called UML Testing Profile for Coordination (UTP-C). UTP-C was created from testing experiences of several web and desktop applications in the Software Engineering Lab, located at the Pontifical Catholic University of Rio de Janeiro. In order to illustrate the use of UTP-C, the paper presents tests modeled for validating an e-commerce multi-agent system.**

Keywords-UML testing profile; model based test; software testing.

I. INTRODUCTION

Creating and executing software tests is an activity that is extremely important in the development process. Depending on the size and complexity of the system evaluated, System Under Test (SUT), a large number of tests should be created and maintained. The U.S. National Institute of Standards and Technology (NIST) informs that systems without adequate tests generate annual costs of up to US$ 59.5 billion [24]. This is almost 1% of the gross domestic product of the U.S.

In order to control software tests, it is necessary to apply a process of management, which makes it possible to execute these tests to evaluate if each one is behaving as expected. Several concerns are identified in this process, such as high costs to recruit or train people, the defining of documentation standards, etc.

One approach that has gained prominence to document and assist the activities of test creation, execution and maintenance is the application of test modeling languages, which provides a graphic view that facilitates the abstraction of concepts and the communication between stakeholders. In the literature, there are several approaches related to test modeling, such as the UML Testing Profile [1], the AGEDIS Modeling Language [2], and the Unified Testing Modeling Language [3].

Over the past six years, the Software Engineering Lab (LES) at the Pontifical Catholic University of Rio de Janeiro has worked extensively on coordinating and carrying out tests of large-scale software systems developed (for web and desktop) for different domains (e.g., petroleum, e-commerce, etc). Based on this experience and a request from a client, who wanted to have all the tests modeled, we investigated how UML could be used to model relevant test data and hence to help the coordination of test execution. These data, which could be modeled, were identified from different sources: (i) test maturity models (TMM [14] and TMMi [15]); (ii) continuous integration tools [16] (e.g., Hudson, Continuum and Cruise Control); (iii) test management tools (e.g., Rational Quality Manager [19] and Rational Test Manager [20]); (iv) test modeling languages; and (v) IEEE documents (such as, IEEE 829-2008 [21]). Some of the identified data were described in [23].

From this work, a test group of the LES proposed a new test modeling language called UML Testing Profile for Coordination (UTP-C), which is presented in the paper. UTP-C is an extension of the UML Testing Profile, which is an OMG pattern for the UML language. This approach was provided to allow the modeling of useful data that help the coordination of software testing. According to Baker e al. [1], a profile defines new stereotypes, attributes, and methods to provide additional semantics for the UML.

When UTP-C was being created, we identified the possibility of generating a set of useful artifacts from UTP-C models. However, to conduct this generation, an appropriate tool needed to be created and used. The artifacts identified for automatic generation were: (i) javadoc commentaries in test script source code; (ii) reports that provide important data about modeled tests; and (iii) a set of XML files considered as input data for multi-agent systems [22] that use the Java Self-Adaptive Agent Framework for Self-Test (JAAF+T) [5][6].

JAAF+T is a framework that aims to allow the creation of self-adaptive software agents that perform a self-test before executing self-adapted behaviors. We consider self-test as the action of validating some adaptation before using it. These validations are performed by a set of tests described in XML files and that are explained in detail in [5] and [6]. Hence, from the JAAF+T, a self-adaptive agent can coordinate the execution of tests, i.e., choosing and executing which tests will validate some self-adaptation performed by it.

Since different LES projects use the Rational Software Architecture (RSA) tool to model UML diagrams, we decided to create a new plug-in for the tool called "RSA applying Model-Based Test" (RSA-MBT). The main focus of this plug-in is to generate test artifacts from UTP-C models.

Thus, the paper is organized as follows. In Section II, the new UML profile is explained. In Section III, a case study is presented that illustrates examples of UTP-C diagrams at an e-commerce multi-agent system developed for the web. These diagrams are modeled using the Astah tool [13]. In Section IV, the main idea of the RSA-MBT plug-in is presented, and the diagrams modeled from the Astah (in Section III) are modeled in the RSA tool. Thus, it is possible to see the modeling based on UTP-C in two different tools. In Section V, conclusion and future works are presented.

II. UML Testing Profile for Coordination

In this section, the UML Testing Profile for Coordination (UTP-C), which was created to model useful data to test coordination, is presented. As stated previously, UTP-C is an extension of the UTP, a standard test profile of the OMG for the UML language. UTP-C uses UML class and activity diagrams for modeling a set of test data. These diagrams were chosen because they allow the modeling of structural and dynamic information that helps the coordination of tests.

The meta-class diagram illustrated in Figure 1 presents a set of stereotypes defined by the UTP-C profile, as well as where they can be used in UML elements. Some of these stereotypes are new, while others are provided by the UTP, but had constraints and properties included. In spite of these inclusions in the UTP-C, they do not challenge the compatibility to the ones that use UTP. Due the limited space of the paper, we will not be able to present in detail these constraints and properties that are described in [6]. However, the example presented in Section III illustrates how UTP-C diagrams can be modeled.

Below, the description of each stereotype used by the UTP-C is presented.

- <<TestCase>>: It states a test case of a system under test (SUT). Each test case is composed of a set of data: test type (e.g., white box, functional, non-function, regression, etc), priority of execution, version of the SUT that it is currently updated, type of obligatoriness, i.e., if execution is mandatory or optional, and the related risk of the system when the test case fails (e.g., to stop the system, data inconsistency, etc.). This set of data related to each test case was not considered by the UTP.

- <<TestContext>>: It states that a set of test cases is responsible for testing some artifacts of the SUT. A test context is composed for: 1 to N test cases, it informs the version of the SUT that their test cases should be updated (desired version), test tool used for executing it, test level related (e.g., unit, integration, system or acceptance), and if it is executed automatically or manually. All these data, except the definition that a test context is composed for 1 to N test cases, were not considered by the UTP.

- <<OrderedSuite>>: It is used to represent a test suite, i.e., an entity that executes a set of test contexts and test cases upon a specific order. UTP considers that a test context is a suite. However, to allow a better identification of a suite class that does not have developed test cases, in comparison to a class that has test cases (test context), we decided to offer the <<OrderedSuite>> stereotype.

- <<TestCriterion>>: It defines a criterion of selection to execute tests of the SUT. An example of a criterion is to execute all the regression and unit tests with high priority and mandatory.

- <<ArtifactUnderTest>>: This stereotype is responsible for representing a set of data related to some artifacts under test (AUT) that are provided in a comment entity. Examples of provided data are the following: path where the results of the tests executed to validate the AUT are stored (result's log), name of the AUT, and type of artifact tested (e.g., class, agent of software, web-service, etc.).

- <<TestClassification>>: It represents a test classification. Test classification is any information that allows grouping and relating test contexts and ordered suites. Its focus is to help the visualization of test entities and their conceptual relations.

- <<Development>>: It represents the real package that stores a given created and modeled class. This is different than the stereotype <<TestClassification>>, which represents conceptual views.

In Figure 1, the Element meta-class is a superclass of the *Classifier* meta-class, which is a superclass of the *Class* (used in class diagrams) and *Activity* meta-classes (used in activity diagrams) [4]. Thus, the TestContext, OrderedSuite, and TestCriterion stereotypes can be used in any sub meta-class of Classifier, while the TestCase stereotype is related to the Behavior meta-class to allow the modeling at behavioral entities, such as Activity.

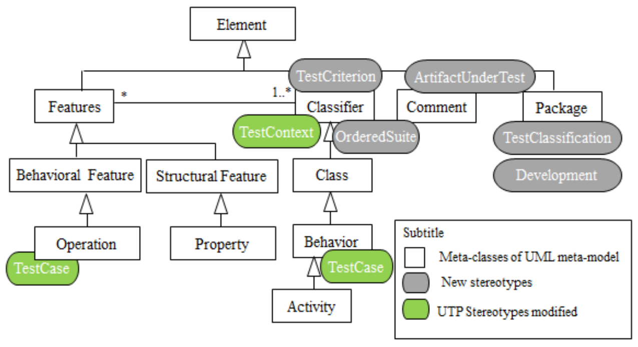


Figure 1. UTP-C meta-model.

Figure 1 also illustrates that the *Classifier* meta-class is related to *StructuralFeature and BehavioralFeature* meta-classes. A structural characteristic is a characteristic of a classifier that specifies the structure of instances of the *StructuralFeature* meta-class, whereas a behavioral characteristic is a characteristic that specifies an aspect of behavior of theirs instances. Thus, the *StructuralFeatures* meta-class is a generalization of *Property* meta-class (attributes of a class are represented as instances of *Property*), and the *BehavioralFeatures* meta-class is a generalization of the *Operation* meta-class, according to the definition of the UML [4]. The original UTP considers that a test case also can be represented as an operation. Hence, the *TestCase* stereotype can be used in the *Operation* meta-class.

The *Comment* meta-class is a subclass of the *Element* meta-class and it can receive the ArtifactUnderTest stereotype. As stated previously, this stereotype informs that data which compose a Comment instance are related to an artifact of the SUT.

TestClassification and Development stereotypes are used in packages (represented by the *Package* meta-class) that allow, respectively, test classifications or development packages to group test contexts and/or suites.

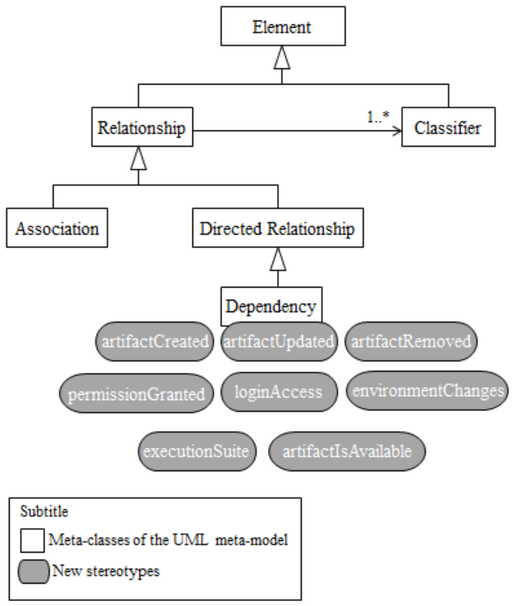


Figure 2. Meta-model of relationships.

Another important data for test coordination is to understand which dependences exist between tests. In order to represent additional semantics on relationships of dependency, a set of stereotypes were proposed by the UTP-C to the UML. These stereotypes were proposed from situations identified in test projects of the Software Engineering Lab. Although this is a limited set, other stereotypes can be included depending on the needs of each project, such as proposals that express more situations of security in SUTs (e.g., <<permissionRevoked>>).

These stereotypes are presented in Figure 2 and described below.

- <<artifactCreated>>: It is used when a test case depends on the creation of some artifact (e.g., file, component, entity, etc.) performed by another test case.
- <<artifactUpdated>>: It states that a test case depends on the updating of an artifact (e.g., changing the name, path, etc.).
- <<artifactRemoved>>: It indicates that the test case depends on the exclusion of another system artifact.
- <<environmentChanges>>: The test case depends on changes in the environment where it is being executed, such as changes to the operating system, environment variables, etc.
- <<permissionGranted>>: It is used when a test case depends on a permission granted from another test case.
- <<loginAccess>>: It states that a test case depends on a login performed in the SUT from another test case.
- <<executionSuite>>: It informs which test contexts an OrderedSuite executes.
- <<artifactIsAvailable>>>: It is used when a test case needs to use an artifact provided by another test case.

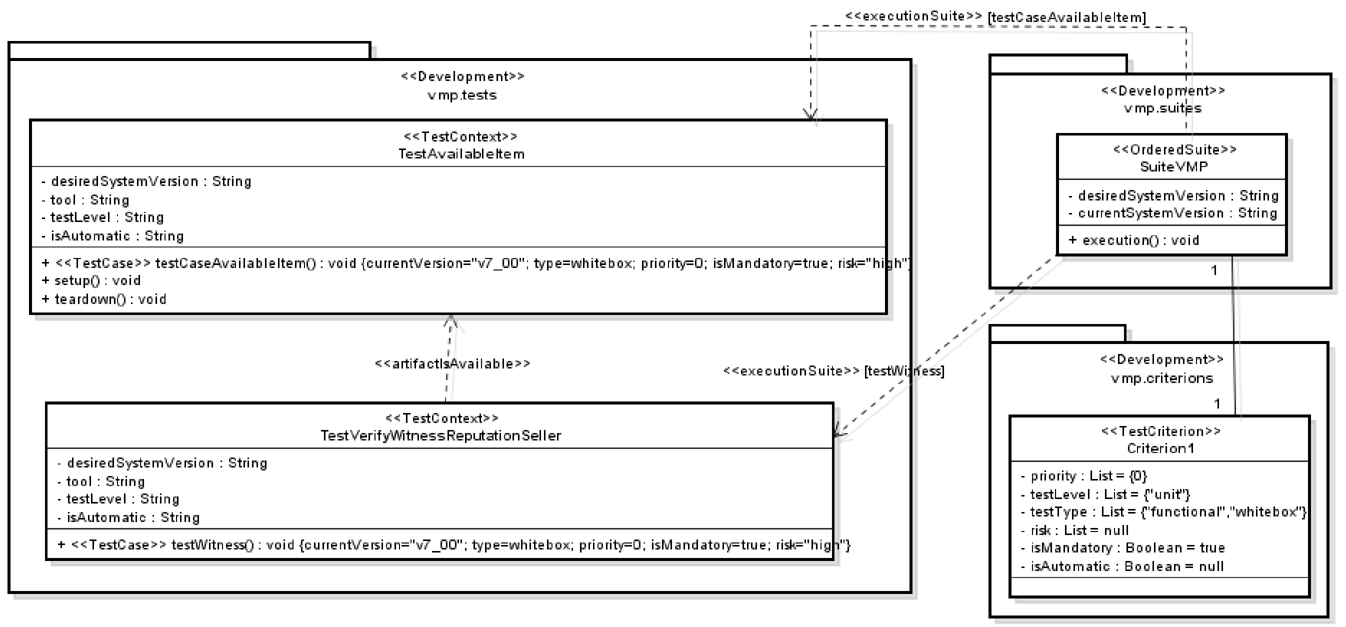
III. Case Study: Virtual Market Place System

This section presents the test modeling of the Virtual Marketplace (VMP) application, an e-commerce system where software agents represent users (buyers) and markets (sellers) that sell new and used books. Each buyer agent executes a set of tests to decide which seller will be used to buy his desired books. In order to show how the UTP-C approach can be used a subset of tests created and executed by the buyer agents are modeled. Thus, this section is organized as follows. In Section A, the idea of the VMP system is presented in more detail, and in Section B, UTP-C diagrams are presented and described.

A. Main Idea

Aiming to exemplify the use of the UTP-C, we decided to use the VMP system that provides markets responsible for selling new and used books for users. As stated previously, each user is represented by a buyer software agent, which negotiates with seller agents that represent markets (e.g., Amazon, Ebay, etc.).

Initially, a buyer user should register with the system providing: (i) its preferred market; (ii) the minimum reputation a seller (market agent) must have; and (iii) if he prefers to buy either new or used books. These data are used by the buyer agent to negotiate with seller agents that satisfy the requests made by the user.



<<executionSuite>> [testCaseAvailableItem]
<<Development>>
vmp.tests
<<TestContext>>
TestAvailableItem
- desiredSystemVersion : String
- tool : String
- testLevel : String
- isAutomatic : String
+ <<TestCase>> testCaseAvailableItem() : void {currentVersion="v7_00"; type=whitebox; priority=0; isMandatory=true; risk="high"}
+ setup() : void
+ teardown() : void
<<artifactIsAvailable>>
<<executionSuite>> [testWitness]
<<TestContext>>
TestVerifyWitnessReputationSeller
- desiredSystemVersion : String
- tool : String
- testLevel : String
- isAutomatic : String
+ <<TestCase>> testWitness() : void {currentVersion="v7_00"; type=whitebox; priority=0; isMandatory=true; risk="high"}
<<Development>>
vmp.suites
<<OrderedSuite>>
SuiteVMP
- desiredSystemVersion : String
- currentSystemVersion : String
+ execution() : void
1
<<Development>>
vmp.criterions
1
<<TestCriterion>>
Criterion1
- priority : List = {0}
- testLevel : List = {"unit"}
- testType : List = {"functional","whitebox"}
- risk : List = null
- isMandatory : Boolean = true
- isAutomatic : Boolean = null

Figure 3. Class Diagram based on UTP-C.

After registering, the user can request the purchase desired. However, a set of data must be provided: (i) title(s) of book(s) desired, (ii) name(s) of author(s), and/or (iii) the maximum price he is willing to pay for book. From these data, the buyer agent (representative of the user) verifies if the seller agent (representative of his preferred market) can meet the request that has been made.

If a seller cannot satisfy the request, the buyer agent tries to meet another seller agent that can sell the desired books. In order to meet another seller, three verifications are performed: (i) if the prices of the desired books provided by the seller are lower than the maximum price informed by the buyer, (ii) if the type of book (used or new) informed by the buyer is respected, and (iii) if the seller agent's reputation is higher than or equal to the minimum reputation of the buyer.

The idea of reputation used on the VMP system is based on the interaction and witness reputations proposed by the Fire model [7]. The interaction reputation is related to the provision of reputations from the negotiation between two agents. In this case, a buyer agent can define a reputation of the seller agent involved in the interaction performed. This reputation is stored in a private buyer agent database. On other hand, the witness reputation allows an agent A to request the reputation (opinion) for an agent B about an agent C. Thus, a buyer agent can request opinions about a seller agent for other buyer agents.

When a seller agent is able to meet the request provided by a buyer, the VMP system presents details of the purchase for the user and it expects confirmation to conclude the negotiation between the agents.

B. Modeling VMP

Figure 3 illustrates a class diagram, created from the Astah tool [13]. This diagram has two test contexts created for the VMP system: TestAvailableItem and TestVerifyWitnessReputationSeller. TestAvailableItem has a test case named testAvailableItem, while TestVerifyWitnessReputationSeller has the test case testWitness. These test contexts execute automatic (use of the attribute isAutomatic) test cases for the version 7.0 of the SUT (represented by the attribute desiredSystemVersion). Furthermore, they use the JAT tool (represented by the attribute tool) [8], which allows the development of unit tests (use of the attribute testLevel) for multi-agent systems.

The main goal of the TestAvailableItem test context is to verify if a seller agent can sell a given book requested by the buyer agent while the TestVerifyWitnessReputationSeller test context verifies if the seller agent has a reputation higher than the reputation informed by the buyer. This conclusion is achieved from the average generated by the reputations provided for other buyer agents of the system about the analyzed seller agent.

Figure 3 shows that each test case of the system contains five more pieces of important associated information: (i) the system version with which the current test case is associated and updated (described by using the attribute currentVersion); (ii) its type of test (e.g., functional, non-functional, regression, etc.); (iii) the priority of the execution (e.g., high, medium, low); (iv) the type of obligatoriness (e.g., mandatory or optional); and (v) the risk related to the test when this test fails. The model allows a description of a risk in detail (e.g., to stop the system) or only its severity related to the SUT (e.g., high severity as illustrated in Figure 3) when a test case to fail. The works presented in [6] and [23] describe in detail the relevance of modeling these test data.

The SuiteVMP class is a test suite responsible for executing the test contexts mentioned previously. If a suite executes a subset of test cases developed through some test context, the modeling can inform which are these test cases from the following structure: <<executionSuite>> [name_test_case_1, …, name_test_case_N].

The entities modeled in Figure 3 are grouped in packages that have the stereotype <<Development>>. This stereotype represents the package where the classes of a given project are stored. On the other hand, the stereotype <<TestClassification>> can be used to group conceptually test contexts and suites. Packages with this stereotype do not store developed classes, different than packages with the stereotype <<Development>>.

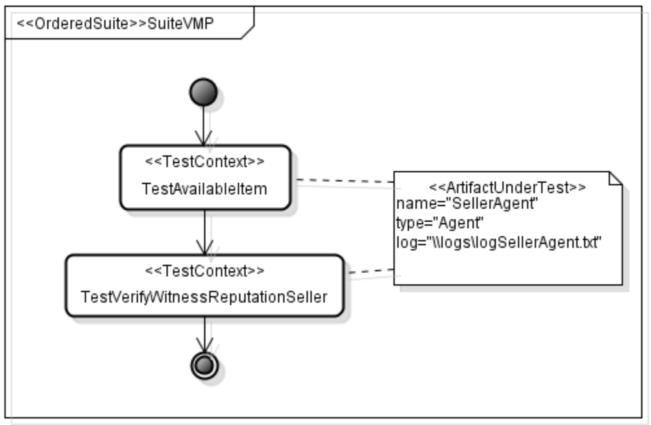


Figure 4. Example of activity diagram.

Finally, but not least important, Figure 4 shows an activity diagram that illustrates the order of execution considered by the SuiteVMP. In this diagram, the first test context to be executed is TestAvailableItem followed by TestVerifyWitnessReputationSeller. The diagram shows that these test contexts are responsible for testing a given seller agent, and the test results are stored at "\\logs\logSellerAgent.txt". These data are provided for a commentary entity with stereotype <<ArtifactUnderTest>> illustrated in Figure 4.

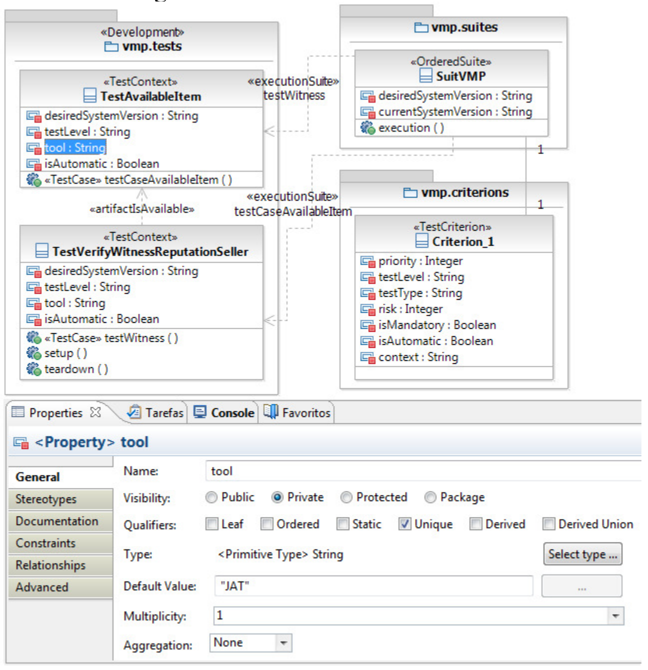
IV. RSA-MBT PLUG-IN

When UTP-C was being created, we identified the possibility of generating a set of useful artifacts from UTP-C models. Thus, the RSA-MBT was proposed. It is an open-source plug-in, developed in Java, for the Rational Software Architecture (RSA) tool, and it is available in [9].

From the RSA-MBT it is possible to generate test artifacts based on UTP-C diagrams. The possible test artifacts, which can be generated from it, are the following: (i) test reports for test teams; (ii) javadoc commentaries; and (iii) a set of XML files used in multi-agent systems that instantiate the JAAF+T framework. Notice that currently the plug-in is not creating test codes. However, we intend to include this generation in the next releases of the plug-in. Thus, the main idea of the RSA-MBT is to generate a set of artifacts that can help the work of test teams, such as understanding characteristics of each test case (e.g., from javadoc commentaries), and knowing which tests are not updated to a specific version of a system under test (e.g., using test reports generated).

The RSA tool allows several transformations, such as from UML diagrams to Java. When this transformation is requested, the RSA-MBT is executed.

Figure 5 illustrates the same classes modeled in Figure 3, but modeled from the RSA tool. Data of the test cases (methods) are presented in the Documentation tab, when a test case method is selected, as illustrated in Figure 6. This approach was considered, because RSA tool does not allow modeling these data of the test cases as the Astah tool. Besides, we informed that the current version of the testWitness is v6_00, which is different from the one in Figure 3. This was performed in order to show better some data generated from the plugin proposed and explained more in the following.



<Property> tool

Name: tool

Visibility: Public [✓ Private] Protected Package

Qualifiers: Leaf Ordered Static [✓ Unique] Derived Derived Union

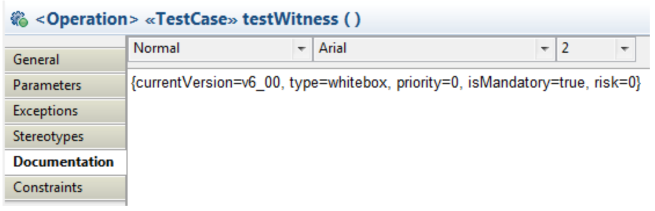
Type: <Primitive Type> String

Default Value: "JAT"

Multiplicity: 1

Aggregation: None

Figure 5. Example of class diagram based on UTP-C.



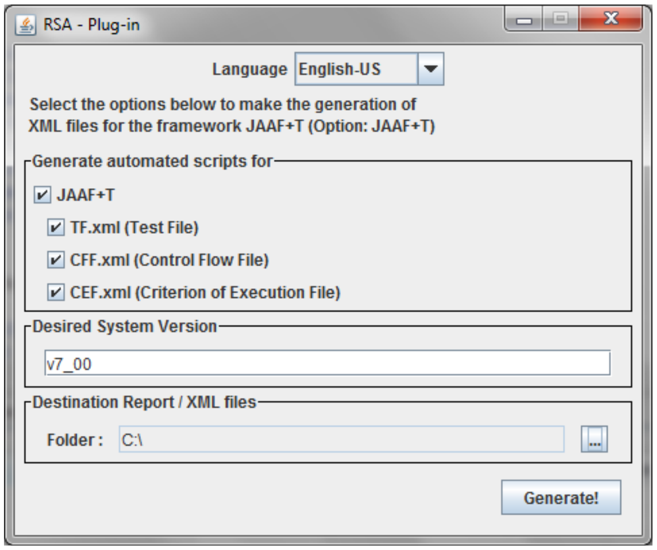
<Operation> «TestCase» testWitness ( )

Normal | Arial | 2

{currentVersion=v6_00, type=whitebox, priority=0, isMandatory=true, risk=0}

Figure 6. Documentation tab of the RSA tool.

From modeling of diagrams based on the UTP-C approach, the user should request the UML to Java transformation. With this request the main screen of the RSA-MBT is presented (see Figure 7). Such a screen allows choosing which test artifacts will be generated and which language must be considered. Nowadays, the plug-in allows generating artifacts in six different languages: English, Portuguese, Italian, French, Spanish, and German.



RSA - Plug-in

Language English-US

Select the options below to make the generation of XML files for the framework JAAF+T (Option: JAAF+T)

Generate automated scripts for

- [✓] JAAF+T
- [✓] TF.xml (Test File)
- [✓] CFF.xml (Control Flow File)
- [✓] CEF.xml (Criterion of Execution File)

Desired System Version

v7_00

Destination Report / XML files

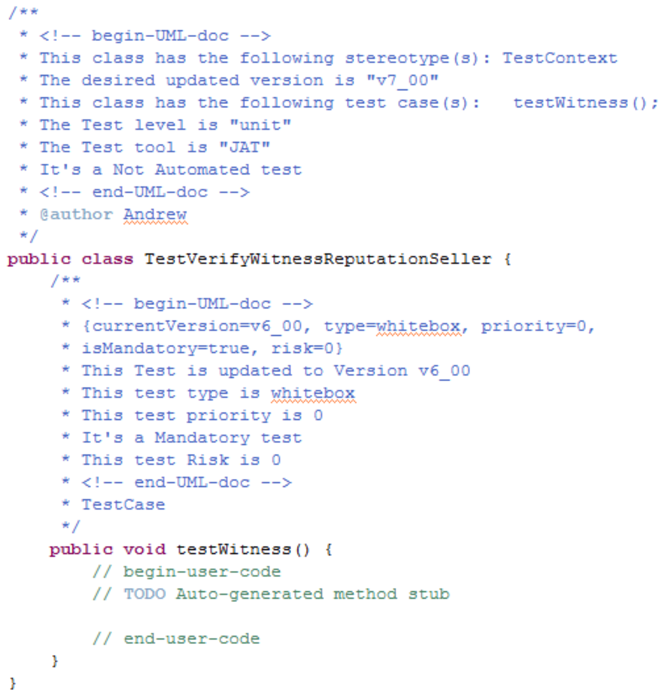
Folder : C:\

Generate!

Figure 7. RSA-MBT screen.

Figure 8 illustrates an example of javadoc commentaries generated in English. In this example, commentaries are provided to the class (test context) TestVerifyWitnessReputationSeller and to its test case (testWitness method) modeled in the class diagram presented in Figure 5.

The commentaries generated to the class TestWitness are based on the data provided in the modeled attributes: desiredSystemVersion, testLevel, tool and isAutomatic. Hence, it is informed that such test context uses the JAT tool, is an automatic and unit test context, and should be updated to the version "v7_00" of the SUT. On other hand, the commentaries generated by the "testWitness" method are based on the data provided in the "Documentation" tab presented in Figure 6. Thus, RSA-MBT informs that it is a mandatory and a white-box test case currently updated to version v6_00 of the SUT. Besides, it has priority and risk 0 (zero), i.e., high priority and risk, respectively.



```
/**
 * <!-- begin-UML-doc -->
 * This class has the following stereotype(s): TestContext
 * The desired updated version is "v7_00"
 * This class has the following test case(s):   testWitness();
 * The Test level is "unit"
 * The Test tool is "JAT"
 * It's a Not Automated test
 * <!-- end-UML-doc -->
 * @author Andrew
 */
public class TestVerifyWitnessReputationSeller {
    /**
     * <!-- begin-UML-doc -->
     * {currentVersion=v6_00, type=whitebox, priority=0,
     * isMandatory=true, risk=0}
     * This Test is updated to Version v6_00
     * This test type is whitebox
     * This test priority is 0
     * It's a Mandatory test
     * This test Risk is 0
     * <!-- end-UML-doc -->
     * TestCase
     */
    public void testWitness() {
        // begin-user-code
        // TODO Auto-generated method stub

        // end-user-code
    }
}
```

Figure 8. Example of javadoc commentaries.

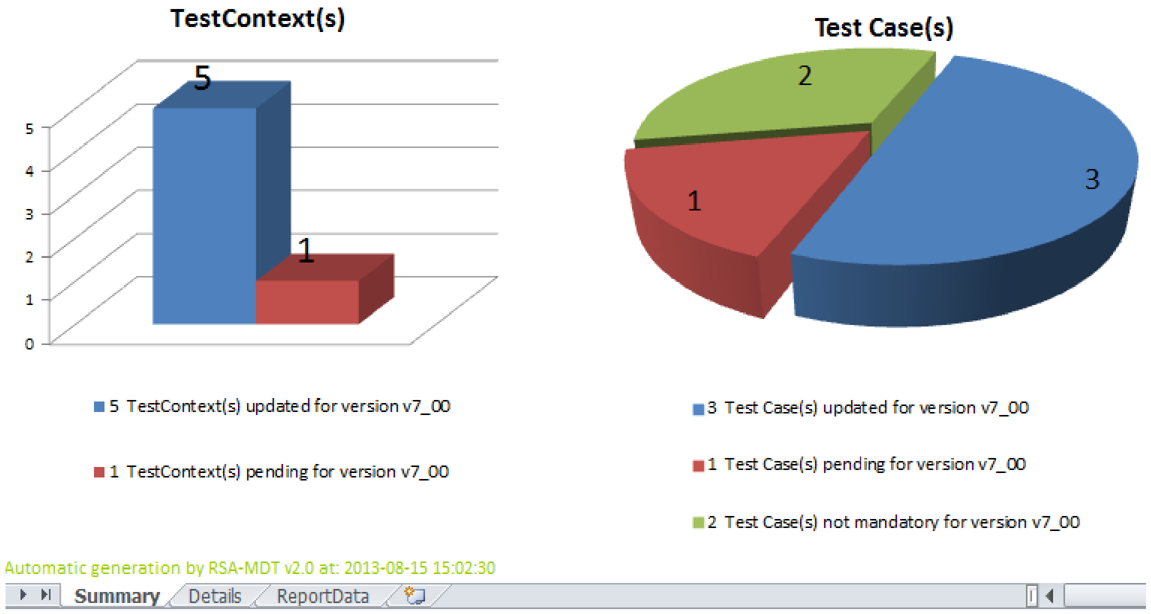
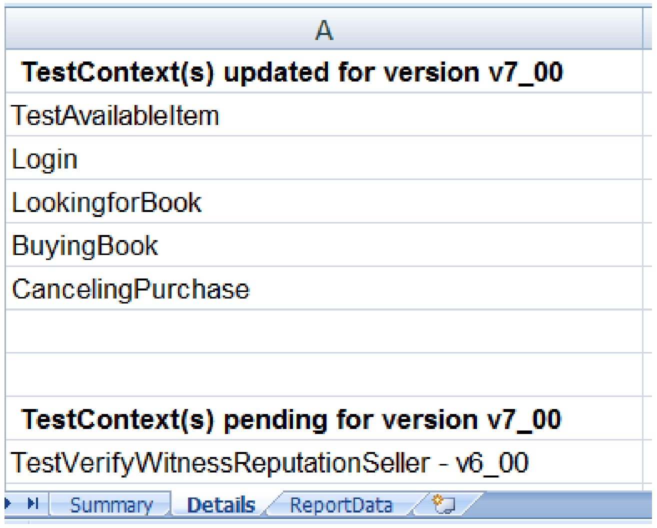


Figure 9. Summary tab – test report generated.

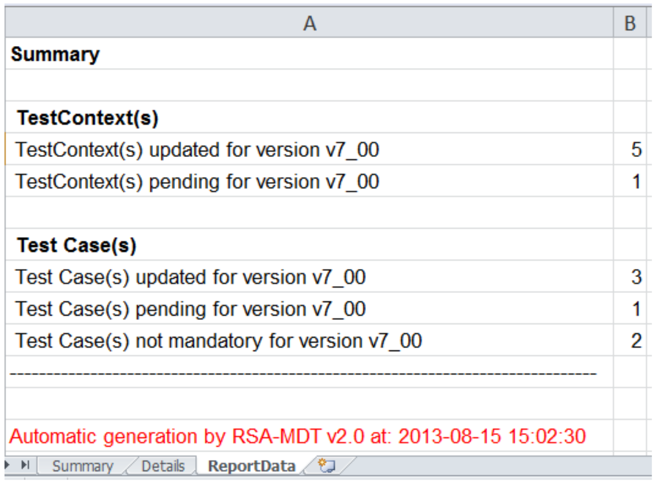
In order to provide an overview of which test contexts and test cases are updated to a specific version informed by the user (by using the text field “Desired System Version” illustrated in Figure 4), a test report (“.xls” extension) is created. This report has three tabs, which are explained in detail as follows.

- **Summary tab** (see Figure 9): It presents two graphics that inform the number of test contexts and test cases updated to the version provided by the user (we are considering that the desired version is v7_00).
- **Details tab** (see Figure 10): It lists the test contexts (test classes) updated and not updated to the version desired.
- **ReportData tab** (Figure 11): It presents an overview of the current state of these updates.



A
TestContext(s) updated for version v7_00
TestAvailableItem
Login
LookingforBook
BuyingBook
CancelingPurchase
TestContext(s) pending for version v7_00
TestVerifyWitnessReputationSeller - v6_00

Figure 10. Details tab – test report generated.



A	B
Summary	
TestContext(s)	
TestContext(s) updated for version v7_00	5
TestContext(s) pending for version v7_00	1
Test Case(s)	
Test Case(s) updated for version v7_00	3
Test Case(s) pending for version v7_00	1
Test Case(s) not mandatory for version v7_00	2
--	
Automatic generation by RSA-MDT v2.0 at: 2013-08-15 15:02:30	

Figure 11. ReportData tab – test report generated.

Also, RSA-MBT generates XML files as input data to the JAAF+T framework. As stated previously, JAAF+T is a framework that allows creating self-adaptive agents that perform self-tests based on a set of XML files.

Three XML files can be generated by the plug-in: TF.xml, CFF.xml and CEF.xml. Test File (TF.xml) is responsible for describing all the tests that can be executed in self-adaptations (see Figure 12). Control Flow File (CFF.xml) presents the order of execution that tests must be executed to validate some artifact of the SUT (see Figure 13). While Criterion of Execution File (CEF.xml) describes the criterions that define which tests, present in the TF.xml file, will be executed (see Figure 14).

```
<tf>
 <test id="testAvailableItem" isAutomatic="true"
        testLevel="unit" testType="whitebox"
        isMandatory="true" context="VMP v1.0">
     <classpath>
     vmp.tests.TestAvailableItem
     </classpath>
     <priority>0</priority>
     <risk>0</risk>
     <tool>JAT</tool>
 </test>
 <test> ... </test>
</tf>
```

Figure 12. Example of a TF.xml file.

```
<cff>
  <artifact id="SuiteVMP" type="Agent"
      logPath="C:\log.txt"
      criterionID="Criterion1">
      <test type="test"
        id="testAvailableItem"/>
      <test type="test"
        id="testVerifyWitnessReputationSeller"/>
  </artifact>
</cff>
```

Figure 13. Example of a CFF.xml file.

The main idea of using UTP-C models was to make creation and maintenance of these XML files easier since, depending on the size of an XML file, the editing work can be difficult. Thus, as all the data considered by the XML files can be modeled in UTP-C diagrams, it is often easier to edit diagrams than to work with XML files. Details of these files are presented in [5].

V. Discussion

One of the most relevant work related to test modeling is the UML Testing Profile [1] that defines a profile for designing, visualizing, and documenting the artifacts of test systems. Such an approach extends UML 2.x [4] with test specific concepts, such as test components, verdicts, defaults, etc. These data are grouped in test architecture, test data, test behavior and time. Being a profile, the UML testing profile seamlessly integrates into UML: it is based on the UML meta-model and reuses UML syntax. Although the approach proposes interesting concepts for modeling test systems, it does not support the modeling of important test data represented by our test modeling language, such as the identification of (i) the system version that each test is able to test, (ii) the mandatory and optional tests, (iii) the test types created, (iv) the types of dependences that exist between the tests (such as data dependence), and (v) the automated and manual tests. On the other hand, the UTP-C approach provides support to represent these test data.

```
<cef>
  <criterion id="Criterion1">
      <priority>0</priority>
      <testLevel>unit</testLevel>
      <testType>functional</testType>
      <testType>whitebox</testType>
      <isMandatory>true</isMandatory>
  </criterion>
</cef>
```

Figure 14. Example of a CEF.xml file.

AGEDIS modeling language (AML) [2], which is another testing language, is based upon the UML (1.4) meta-model and enables the specification of tests for structural (static) and behavioral (dynamic) aspects of computational UML models. AML comes as part of the AGEDIS methodology and has been designed with two main goals in mind: to create a test adequate abstraction of the SUT that will be analyzed by the AGEDIS tools, which allows generating automatically suite tests, and to set meaningful test directives for the testing process. AML presents the same problems mentioned for the UML Testing Profile.

The Testing and Test Control Notation (TTCN-3) [11] is a modular language that has the similar look and feel of a typical programming language. This language is widely accepted as a standard for test system development in the telecommunication and data communication area. The main reason for such acceptance is that it comprises concepts suitable to any type of distributed systems to be tested, such as important features necessary to specify test procedures for functional, conformance, interoperability, load and scalability tests. Besides this, it defines mechanisms to compare the reactions of the system under test with the expected range of values, time handling, distributed test components, ability to specify encoding information, synchronous and asynchronous communication, and monitoring. Similar to the UML Testing Profile, TTCN-3 also does not provide a set of useful concepts that the test modeling language, presented in this paper, proposes. All the concepts not included in the UML Testing Profile and AGEDIS are also not considered in this work.

According to [3], the benefits of Model-Driven Engineering (MDE) for product software development have been demonstrated in numerous instances. Therefore, similar benefits can also be achieved in applying MDE to test software development. This form of Model-Based Testing (MBT) is called Model-Driven Test Engineering (MDTE) or simply Model-Driven Testing (MDT). However, to optimize the efficiency of MDT, good-practices and patterns specific to test development must be taken into account. Based on this idea, Feudjio [12] proposes a Unified Test Modeling Language (UTML) that is a test notation designed for pattern-oriented MDT. It provides the means for designing all aspects of a test system at a high level of abstraction and independent of any specific lower-level test infrastructure. Besides this, at the same time it provides guidance in following test design patterns and avoids usual pitfalls of MDT. Such an approach provides a tool called MDTester that allows modeling the concepts proposed by UTML. However, this tool does not allow to explicitly model the test data provided by the UTP-C, such as, test type, test level, risk, priority, etc.

VI. Conclusion and Future Work

This paper presented a new test modeling approach named UML Testing Profile for Coordination (UTP-C). This approach extends the UML Testing Profile in order to model useful data that helps test coordination. These data were identified from tests created and executed for different systems (web and desktop) in the Software Engineering Lab. This work has been motivating research related to the test area, especially the Model Based Test, such as the creation of the RSA-MBT plugin, presented in the paper.

Considering that the plug-in was created for the Rational Software Architecture (RSA), when a transformation is requested in the RSA, files generated by the tool are replaced (e.g., Java files created from UML diagrams). Due to this behavior, we are currently developing a treatment that allows applying a merge between Java files. Thus, important contents of Java files already created will not be lost when a UML to Java transformation is requested.

Besides, we are deciding how to automatically generate codes for test scripts for the Rational Functional Tester (RFT) [17] and for the Rational Performance Tester (RPT) [18]. RFT and RTP are tools used in different test projects of the LES that allow creating functional and performance test scripts, respectively.

VII. Acknowledgments

This work has been sponsored by the INCT on WebScience through grants from CNPq and FAPERJ.

References

[1] P. Baker, Z. Ru Dai, J. Grabowski, O. Haugen, I. Schieferdecker, and C. Williams, “Model-Driven Testing: Using the UML Testing Profile”, Springer, ed. 2008, December, 2007.

[2] A. Hartman and K. Nagin, “The AGEDIS Tools for Model Based Testing”, Book UML Modeling Languages and Applications, vol. 3297, 2005, pp. 277-280, doi: 10.1007/978-3-540-31797-5_33.

[3] UTML - The Unified Test Modeling Language for Pattern-Oriented Test Design, <http://www.fokus.fraunhofer.de/distrib/motion/utml/>, retrieved: August, 2013.

[4] UML 2 Specification, <http://www.omg.org/spec/UML/2.3/>, retrieved: August, 2013.

[5] A. D. Costa, C. Nunes, V. T. Silva, C. J. P. Lucena, and B. Fonseca, “JAAF+T: A Framework to Implement Self-Adaptive Agents that Apply Self-Test”, in proceedings of the 25th Symposium On Applied Computing, Sierre, Switzerland, 2010, pp. 928-935.

[6] A. D Costa, “Automation of the Management Process of the Test of Software”, Thesis at Portuguese, Pontifical Catholic University of Rio de Janeiro, August, 2012.

[7] T. D. Huynh, N. Jennings, and N. Shadbolt, “FIRE: an Integrated Trust and Reputation Model for Open Multi-agent Systems. In Proceedings of the 16th European Conference on Artificial Intelligence”, Valencia, Spain, 200, pp.18-22.

[8] R. Coelho, E. Cirilo, U. Kulesza, A. Staa, A. Rashid, and C. J. P. Lucena, “JAT: A Test Automation Framework for Multi-Agent Systems”, in Proceeding of the International Conference on Software Maintenance, France, 2007, pp. 425-434.

[9] RSA-MBT: Web site for downloading, <http://www.les.inf.puc-rio.br/escritorioqualidade/index.php?option=com_content&view=article&id=57&Itemid=58>.

[10] Rational Software Architect, <http://www.ibm.com/developerworks/rational/products/rsa/>, retrieved: August, 2013.

[11] TTCN-3 web site, <http://www.ttcn-3.org/>, retrieved: August, 2013.

[12] A. V. Feudjio, “MDTester User Guide”, <http://www.fokus.fraunhofer.de/distrib/motion/utml/>, retrieved: August, 2013.

[13] Astah tool, <http://astah.net/>, retrieved: August, 2013.

[14] I. Burnstein, A. Homyen, R. Grom, and C.R. Carlson, “A Model to Assess Testing Process Maturity”, Crosstalk 1998, Software Technology Support Center, Hill Air Force Base, Utah, <http://www.crosstalkonline.org/storage/issue-archives/1998/199811/199811-Burnstein.pdf>, retrieved: August, 2013.

[15] TMMi: The Test Maturity Model Integration, <http://www.tmmifoundation.org/html/tmmiref.html>, retrieved: August, 2013.

[16] P. M. Duvall, S. Matyas, and A. Glover, Continuous Integration: Improving Software Quality and Reducing Risk, Publisher: Addison-Wesley Professional, 2007.

[17] Rational Functional Tester, <http://www-03.ibm.com/software/products/us/en/functional/>, retrieved: August, 2013.

[18] Rational Performance Tester, <http://www-03.ibm.com/software/products/us/en/performance/>, retrieved: August, 2013.

[19] Rational Quality Manager tool, <http://www-03.ibm.com/software/products/us/en/ratiqualmana/>, retrieved: August, 2013.

[20] Rational TestManager tool, <http://www-01.ibm.com/software/awdtools/test/manager/>.

[21] IEEE 829-2008 – IEEE Standard for Software and System Test Documentation, <http://standards.ieee.org/findstds/standard/829-2008.html>, retrieved: August, 2013.

[22] M. Wooldridge and N. R. Jennings, “Pitfalls of agent-oriented development”, in Proceedings of the Second International Conference on Autonomous Agents (Agents'98), ACM Press, Minneapolis, USA, 1998, pp. 385-391.

[23] A. D. Costa, V. T. Silva, A. Garcia, and C. J. P. Lucena, "Improving Test Models for Large Scale Industrial Systems: An Inquisitive Study", in Proceedings of the ACM/IEEE 13th International Conference on Model Driven Engineering Languages and Systems, Part I, LNCS Springer 6394, Oslo, Norway, 2010, pp. 301-315.

[24] NIST: National Institute of Standards and Tecnology, Software Errors Cost U.S. Economy $59,5 Billion Annually – NIST Planning Report 02-3, 2002.

An Ontology-based System to Support Distributed Software Development

Rodrigo G. C. Rocha, Ryan Azevedo, Eduardo Tavares, Daniel Figueredo
UFRPE
Garanhuns – PE, Brazil
rodrigo, ryan@uag.ufrpe.br, eteduardotavares, jdaniellll3593@gmail.com

Catarina Costa
Department of Statistics and Mathematics
Federal University of Acre
Rio Branco – AC, Brazil
catarina@ufac.br

Marcos Duarte
Information Systems Course
Paraíso College of Ceará
Juazeiro do Norte – CE, Brazil
marcos.duarte@fapce.edu.br

João Paulo Fechine
UNIPETECH
UNIPE
João Pessoa – PB, Brazil
fechine@gmail.com

Fred Freitas, Silvio Meira
Federal University of Pernambuco – CIn
Recife – PE, Brazil
fred,srlm@cin.ufpe.br

Abstract— **Distributed Software Development has become an option for software companies to expand their horizons and work with geographically dispersed teams, exploiting the advantages brought by this approach. However, this way of developing software enables new challenges to arise, such as the inexistence of a formal, normalized model of a project's data and artifacts accessible to all the individuals involved, which makes it harder for them to communicate, understand each other and what is specified on the project's artifacts. With that being said, this paper proposes a knowledge management tool that utilizes a domain-specific ontology for distributed development environments, aiming to help distributed teams overcome the challenges brought by this modality of software development proposing techniques and best practices. Thus, the main output of this work is Ontology-based System to Support the software development process with distributed teams.**

Keywords-Distributed Software Development; Ontologies; Knowledge.

I. INTRODUCTION

Motivated by opportunities like the availability of experts worldwide, cost reduction, local government incentives and employee turnover reduction, several software development companies have been starting to work with geographically distributed development teams, adopting the Distributed Software Development approach.

The aforementioned distribution of teams brings along with it new challenges to the software development scenario. Carmel [1] and Komi-Sirvo and Tihinen [2] reiterate the existence of these challenges by presenting some factors that are likely to lead distributed software development projects into failure: inefficient communication between distributed team members, diverging cultures and high complexity or lack of project management.

In this context, the nonexistence of a formal, normalized project data model accessible by the entirety of the team makes the communication between them and the understanding of the project artifacts harder, which can be aggravated when each member's culture and customs is barely or even not known by the rest of the team.

In order to mitigate these problems, the utilization of ontologies can be useful because they enable the creation of a common vocabulary. Wongthongtham *et al*. [3] mention that the use of ontologies represent a paradigm shift in Software Engineering and can be used especially to provide semantics for support tools, strong, knowledge-based communication, centralization and information availability.

This paper proposes DKDOnto, a domain-specific ontology for distributed software development projects, whose purpose is to aid those projects by defining a common vocabulary for distributed teams. Besides, this work proposes a tool that enables both handling and searching the information in the knowledge base, in order to get more useful information as to mitigate and avoid future problems inside the project.

The main goal of this work is the proposal of both the ontology and the tool, which together will compose a mechanism to ease the distributed software development process, from sharing of common knowledge between distributed team members or smart agents to the decision-making process effectuated by the project managers.

This paper is organized as follows: Ontology concepts are presented in Section II; Section III contains the knowledge-based system proposal; Related works are presented in Section IV, where a succinct analysis and comparison of related work and this paper is made; and, finally, Section V brings the final considerations.

II. ONTOLOGIES

Various definitions are given as to determine a meaning to ontologies in the Computer Science context, the most popular and best-known definition being "a formal, explicit specification of a shared conceptualization", given by Gruber [4]. By 'formal', he means that it is declaratively defined so that it can be comprehended by smart agents; by 'explicit', he means that the elements and their restrictions are clearly defined; by 'conceptualization', he means an abstract model of a field of knowledge or a limited universe of discourse; by 'shared', he indicates it is consensual knowledge, a common terminology of the modeled field. Thus, ontologies set an unambiguous, common higher abstraction level for several knowledge domains.

Ontologies, according to Guizzardi [5], are composed by concept, relations, function, axioms and instances. In short, concept can be 'anything' about 'something' that is going to be explained. The interaction between a domain's concepts and attributes is called relation, whose type is called function. Axioms model sentences that are always true and instances represent elements from the domain associated with specific concepts.

The use of ontologies has been made popular by many other Computer Science subfields, such as: Software Engineering, Artificial Intelligence, Database Design, and Information Systems. One of the principal persons responsible behind this phenomenon is Web Semantics' creator [6], Sir. Tim Berners Lee.

Many reasons instigate the development of ontologies, according to [7] [8]. Some of these reasons are:

- Sharing common understanding of how information is structured between humans and smart agents;
- Reusing knowledge of a domain. In case there is an ontology that adequately models certain knowledge of a domain, it can be shared and used by engineers and ontology developers, as well as teams that develop semantic and cognitive applications;
- Making explicit assumptions of a domain. Ontologies provide vocabulary to represent knowledge and its use prevents misinterpretations;
- Possibility of translation from and to various languages and knowledge representation formalisms. The translation concretizes an ideal pursued for generations by researchers in Artificial Intelligence. It makes it easier to reuse knowledge, and may allow for communication between agents in different formalisms, since this service is available in an increasing number of knowledge representation formalisms. Another way to reach this intent is to use ontology editors in which it is possible to choose in which language of representation the generated code is going to be written.
- The mapping between two knowledge representation formalisms, that, inspired in the connectivity component for Open Database Connectivity (ODBC) management systems, links two formalisms creating an common access interoperable interface for them, allowing an agent to access the other agent's knowledge.

Furthermore, ontologies help solve some of DSD project problems; for example, how to establish better communication, allow a homogenous comprehension of project information, make the project management a less laborious task, prevent task interpretation errors and synchronize the enrolled, distributed team's efforts and facilitate the knowledge sharing and standardization.

III. Knowledge-based System Proposal

In this work, we present the DKDOnto, a domain-ontology according to classification adopted by [9], which classifies the types of ontologies in: i) generic, ii) domain, iii) task and iv) application.

The ontology proposed intends to be the basis for possible solutions of knowledge-based systems in the context of global software development, in order to assist all the professionals (client too) involved in the software development process with distributed teams. The DKDOnto emerges, thus, as a common knowledge base for this context, leveraging the challenges deals, best practices and possible solutions, as well a road map with all the actors and their assignments.

This proposal takes a step beyond, discussing also an inference engine called DKDs, extremely flexible, customizable for each environment and giving support for the professional in real time. The general flow, operating means and features of the proposed system and the DKDOnto, as well as a systematic mapping study (methodology) are presented in the following subsections.

A. Systematic Mapping Study

In this research, a Systematic Mapping Study was conducted to identify ontologies supporting the DSD. And indirectly to identify tools, techniques, best practices, and models that use ontologies to support this area.

An important issue in this process was to search for reviews and accurate analyses on the field, looking for current researches and open challenges related to the use of ontological resources in Distributed Software Development processes. Thus, the following research question were intended to be answered: "Which ontologies have been proposed or adopted in the context of DSD?"

The searches for the primary studies were conducted according to the research plans defined in the protocol. The search process retrieved 1588 studies from the chosen scientific databases.

This question aims to find out which are the ontologies normalized on the DSD context. In order to answer this research question, four ontologies have been found. Table 1 presents the proposed ontologies in the distributed context. The first column presents the name and identifier of each ontology. The second column shows a description of each one.

Based on results, it is evident that the development phases that are benefiting from the use of ontologies are: process, management, requirements and design. On the other hand, some important branches have not been fully approached, for example, quality and tests, which involves lots of information management activities, and may have a considerable evolution with the utilization of ontologies as means to standardize, manage and share knowledge.

By answering the research question from this mapping, there have been found four works that propose some ontologies, especially developed for distributed software development, according to what was previously presented.

Since these ontologies have been designed specifically for distributed teams, they bear the concepts and features required to work in this environment. Noteworthy to mention that two of the four ontologies were developed for open-source software development communities. According to Mirbel [10], the free dynamic nature of this environment poses challenges to the coordination of activities and knowledge sharing.

Therefore, the use of ontologies as a support to open-source software development simplifies the management of knowledge resources in the communities. Noticeable that several other works use ontologies to solve or mitigate challenges and in DSD environments, however, these ontologies are not specific for this environment.

Thus, four ontologies have been found, they are not shared which does not allow further evaluation and according to the literature, they have not correct modeling to cover the entire software development process using distributed teams. But they have a major limitation, they have not resources to recommend best practices for possible problems.

There are numerous tools that utilize nonspecific-to-DSD ontologies only to mitigate challenges and limitations. These tools are distributed and used as support in the various project parts, from actual Software Engineering branches to specific project activities.

TABLE I. ONTOLOGIES FOR DSD

Models	Description
OFFLOSC[10]	This ontology is formalized in the context of open-source software development communities. Its goal is help coordinate activities, management of resources and knowledge sharing. It is composed by 46 classes and describes the concepts related to open-source communities such as actors, artifacts, activities, operations, relationships and resources.
Knowledge Management Ontologies [11]	A set of ontologies that formalize structural concepts of DSD environments, directed to knowledge management. It describes concepts of software artifacts, environment problems, interaction among the distributed development teammates, infrastructure, business rules and general information of the project.
Open Source Communities [12]	This ontology is also formalized in the context of open-source software development and its main purpose is to compose a project knowledge basis having semantically related, categorized data, which allows the execution of semantic searches and data inferences by smart agents. It is composed of 6 classes that describe concepts of actor's relations, rules, activities, processes, artifacts and tools from open-source communities' projects.
OntoDISEN [13]	This ontology is formalized in the DSD Project scenario and is used to aid the establishment of communication between distributed teams. It is integrated to a textual information-spreading model, enabling sharing information in distributed environments to be comprehended by all the software engineers in a clear, homogeneous way. It describes concepts of elements that are represented and shared in a DSD environment, such as users, tools, other environments, activities and processes.

With these results, it is clear that there are a lot of advantages in using ontologies to support DSD, especially to generate solutions aiming at mitigating the communication, collaboration, knowledge flow management, coordination of project activities and knowledge, and process management issues.

B. *DKDOnto: Proposal Ontology*

The DKDOnto ontology was developed using Ontology Engineering, Methontology [14] and IEEE Standard [15] for developing knowledge-based information systems methodologies; also, Method 101, proposed by N. F. Noy and D. L. Mcguinness's [7] was used a complement to Methontology.

Thus, the language used to build the ontology was OWL, which eases the publication and sharing of ontologies [16] and it has also been proposed as a standard for the World Wide Web Consortium (W3C), incorporating and taking advantage of the strength of earlier languages. OWL is an ontology language (Semantic Web [17]) with high-level expressivity and great potential for knowledge inference. In order to edit the ontology, the use of Protégé [18] was employed. It is a free, extensible, Java-based, open-source ontology editor and knowledge-based framework.

The DKDOnto has about 50 classes, but this paper describes the following core classes.

• Project: the main class of this knowledge base. It is responsible to store all the information about the settings of projects, from allocated team members to phases to activities to artifacts used.

• Member: it is a subclass of Resource. Member is an individual who has access to the environment and are allocated to Projects. A member has skills and works in a place and participates directly in the project, reporting best practices and challenges, using and creating artifacts.

• Best Practices: all the solutions and best practices used to face any problem should be stored in this entity. This class is responsible for helping avoid challenges and problems found and reported by a member during the execution of their activities. It also to solve these challenges and problems.

• Challenges: all the challenges and problems found by members should be stored in this class. A challenge can use best practices to solve itself. This entity is fundamental because the challenges has some solution or best practice associated with some practice can be used and available to another members with same problems.

• Skills: all members' knowledge are stored in this entity. The Member's skill enables to avoid challenges and solve it too. This class allows too that activities be distributed for the members according their skills.

• Place: it is a fundamental class to define exactly where the envolved member are in Project. This entity estores all information about member's localization, defining what is dispersion level and temporal distance.

• Artifact: class that is used by almost all other main classes. It supports members and their activities. Tools can use artifacts in specific activities, too.

• Tool: class reponsible for all the tools envolved in Project. It allows that all the users knows which tools are used for another members and another projects. This way, is possible to follow the patterns and find specific informations and instructions for use this tools.

• Workspace: is a class that contains Artifacts and Tools that Users can use and create in their activities. All the users allocated in that Project can be access the workspace for commit and checkout all the documents, artifacts or tools. The main goal of this class is storable of Artifacts and the Member uses the Workspace of that specific Project.

This ontology uses two fundamental classes for the sucess of this proposal. These classes are responsibles for storage all information about the problems and solutions during the project. These classes are called of Challenges and BestPractices. Thus, user's queries allows to view responses of the challenges, the knowledge base returns the best practices found for a certain team setting and can be applied to support challenges, which can be useful for other teams involved with the same project or other teams from different projects.

Figure 1 shows some relations between classes defined in DKDOnto. This diagram of generated from a plugin for Protégé called Ontoviz. For space constraints, a restrict set classes was chosen to be exhibited.

C. *DKDs: Proposal Tool*

DKDs was developed to aid in the transmission, generation and distribution of knowledge. It is a support tool for decision-making in DSD, which, based in resources and information from the context of a project, the system suggests possible solutions for the problems found to its users. In this sense, the system accesses the knowledge base having distributed projects experiences, their configurations, challenges faced and solutions used to overcome those challenges.

This tool's main goal is to support the complete DSD process, offering recommendations considering the project setting and organization, technical and nontechnical experiences.

In order to develop DKDs, the general platform adopted was J2EE [19]; the web application frameworks utilized were Grails [20] (High-productivity web framework based on the Groovy language [21]) and Google Web Toolkit (GWT) [22]; Hibernate (Java persistence framework project) [23] was used for persistence; and to manipulate the ontology, the Jena framework was employed, which is also responsible for construction and manipulation of Resource Description Framework (RDF) [24] graphics.

With the DKDs a member from a project can know who are the another members envolved and have some instructions to talk each other depending their cultural characteristics. So, it helps to avoid any problems the communication (email, talk, phone). Furthermore, any doubt about some artifact or activity can be solved with the correct member, that is indicated by the tool.

Among DKDs' main features, the most important ones are: DKDs uses the inference engine Pellet for inferring facts based on the information that has been previously stored in the knowledge base, thus, some outcomes that the system can generate:

- Starting the project, request a guideline with suggested best practices for similar contexts
- Starting the project, request a guideline with main challenges for similar contexts
- Determines who are the most qualified members to solve technical problems;
- Suggests possible practices, tools or techniques that can be employed to avoid challenges
- Find possible solutions used previously to problems encountered
- Evaluating the solutions proposed by the tool
- Suggest adaptations to the proposed solutions

The application is basically composed by four modules:

- Inference Module: allows for a precise deduction of information about DKDOnto in RDF and OWL code, using inference engine Pellet.
- Query Module: this is where all the queries made by users occur. As it was mentioned earlier, queries are made in SPARQL language and are transparent to the users.
- Views Module: gives access to all the reports made according to the users' needs.
- Management Module: responsible for enabling access to the ontology with insertion, removal and editing of the data in the ontology permissions.

For example, an user can access the application and insert, delete, edit and view all the data (instances contained in DKDOnto) by the Management Module. The same user can use View Module for the ask the system to inform what is necessary, so, this module activates the Query Module that use the Inference Module to bring appropriate responses for the user.

The users have an access interface to execute the abovementioned functions on one side, whereas on the other side, there is the SPARQL (Query language for Resource Description Framework) [25] inference engine to consult

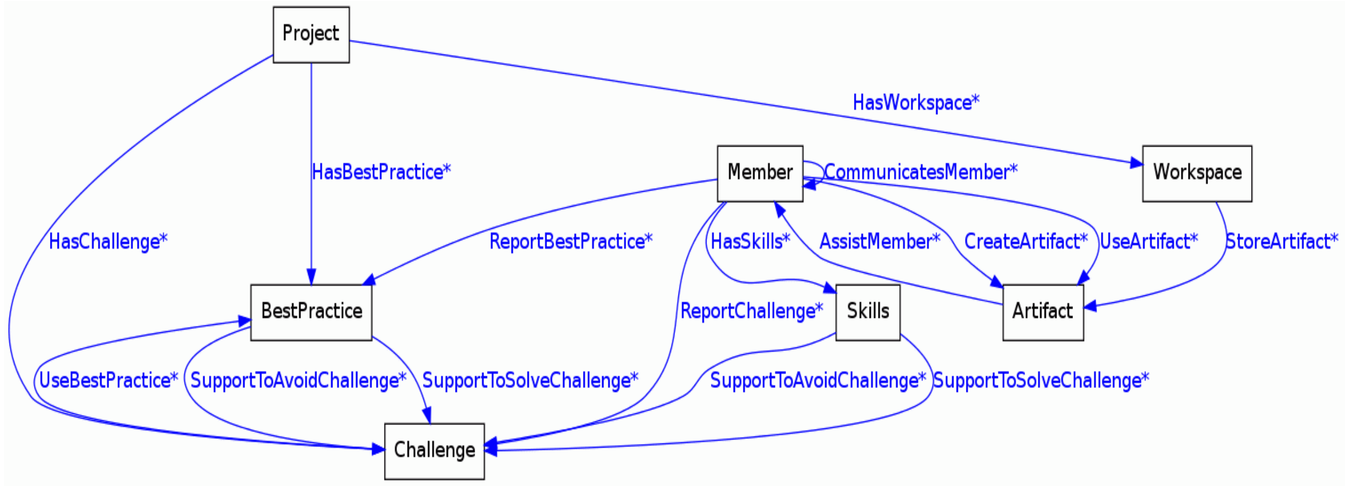


Figure 1. The Core classes and relationships of DKDOnto

DKDOnto, and the interface component (OWL API [26]) in the middle, which interacts with both sides. Integrating all the demands from user using the inference module.

Figure 2 shows the tool's general functioning as described above.

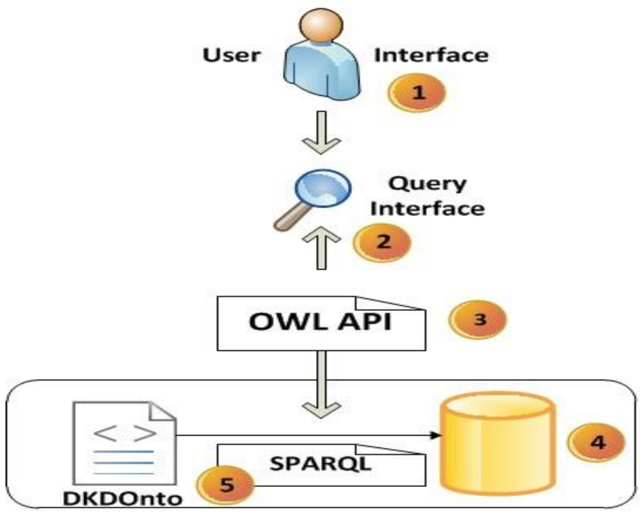


Figure 2. Tools General Functioning

IV. RELATED WORK

In this section, works having the same goal or theme of this paper are described. Based on the amount of related works found, it can be affirmed that relatively few works on Software Engineering Ontologies have been carried out.

Wongthongtham *et al.* [27] present the project and implementation of a social network approach as a mean to support the sharing and evolution of a Software Engineering ontology. A multi-agent recommender system that uses the 'Software Engineering Ontology' and 'SoftWare Engineering Body of Knowledge (SWEBOK)' as sources of knowledge is designed within multi-site communities of software engineers and developers working on related projects as the target audience. Though a big challenge faced by this approach is ensuring that the knowledge bases of different agents are coherent and consistent with one another, as stated by Dilon and Simmons [28].

Ankolekar *et al.* [29] considers as one of the toughest problems faced by online professional communities the fact that the vast amount of data generated as a result of their interactions is not well-linked on the basis of the meaning of its content. With the assumption that a better semantic support can bring improvements to these communities, a prototype Semantic Web system was developed.

Such task required a way of describing the semantic content retrieved from the data obtained from these communities, which was accomplished through the use of ontologies. The large amount of data generated was a large obstacle as the parsers used were unable to reason efficiently for large amounts of data.

The 'instance Store (iS)' system was the solution for such problem, for it stores assertions about individuals and their types in a database, reducing reasoning over individuals to terminological reasoning. But the version of iS used was limited to role-free reasoning of individuals, what at first was deemed to be a major limitation but was dismissed by the authors since the primary use of ontologies in the system "is for the description, annotation and retrieval of large number of individuals" and it "does not make use of the open world assumption nor does it make use of ontologies distributed over multiple sites".

In their work, Dillon and Simmons [28] reiterate the growing importance of the use of ontologies in various aspects of Software Engineering, showing examples ranging from the support that offered to multi-site developers, to the provision of semantics to different categories of software. The 'Software Engineering Ontology' is described and used for the creation of a software engineering knowledge management system that is formed by a 'safeguard system', 'ontology system' and a 'decision-maker system'. The purpose behind this system is to facilitate knowledge sharing, access, update and exchange.

The essential difference of this work is the proposal of the use of best practices for the challenges found by any member, thus, they can be use the DKDs to check or consult all knowledge stored looking for possible best practices. It also allows the creation of a list of possible problems during the initial phases, so the manager or developers can avoid some challenges. Other interesting resource is the creation of a list of possible developers who may be able to help solve technical problems through their skills.

V. CONCLUSION

As globalization took place, the distribution of software development processes have become an increasingly common fact. The DSD work environments are very complex and there are no mature practices for this context since it is relatively new. In this sense, ontologies can bring benefits such as a shared understanding of information, ease of communication among distributed teams and effectiveness in information management.

This work presents evidences from collected papers and a briew analysis of the results reached. The results support the foundation for proposing and developing a feature based on ontologies to support the DSD. The systematic mapping aimed to identify ontologies formalized in DSD context, provided that advance the state of art, highlighting the need to use ontology in this field. Is possible to view all the Systematic Mapping Results in Borge's work [30]. The complete information about it is available at a specific repository files [31].

DKDOnto and DKDs fulfill what has been proposed, consisting of a computing tool that can be used for treatment, analysis and utilization of information on distributed software projects. In this sense, the ontology and the tool allow that actors in this scenario obtain and access correct information and artifacts, providing a high-level knowledge model for the team members.

The results obtained to this date are expressive, in which, for example, the project manager has actual consistent knowledge of which cultures are involved in the distributed teams and which are the implication of this, which enables them to handle each case effectively. Similarly, a technical leader has access to the project participants' technical knowledge, making them able to require or assign specific activities accordingly to the expertise of each team member.

Another important point is that the ontology, as presented in Section 3, has two fundamental classes, namely Challenges and Solutions that are directly utilized by the query tool. That way, the knowledge base will return the challenges found for a certain team setting and also which solutions can be applied to such challenges, which can be useful for other teams involved with the same project or other teams from different projects.

The next step in this segment is to concretize the acquisition of knowledge in a systematic way in order to fill the ontology. In this case, it will be possible to make tests and simulations with higher precision since all the inserted data will be from real projects. Furthermore, other techniques can be used for improves the support of Challenges, for example, the use of natural procesing language for retrieve better solutions or best practices based in challenges cases.

ACKNOWLEDGMENT

Rodrigo was supported partially by CNPQ and Ryan are supported by CNPq. This work is partially supported by the National Institute of Science and Technology for Software Engineering (INES), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08.

REFERENCES

[1] E. Carmel. "Global Software Teams: Collaboration Across Borders and Time Zones". Prentice-Hall, EUA. 1999.

[2] S. Komi-Sirvo and M. Tihinen. "Lessons Learned by Participants of Distributed Software Development". Journal Knowledge and Process Management, vol. 12 no 2, 2005, pp. 108–122.

[3] P. Wongthongtham, E. Chang, T. Dillon, and I. Sommerville. "Ontology-based Multi-site Software Development Methodology and Tools". J. of Systems Architecture. ACM, New York. 2006. 640–653.

[4] T. Gruber. "Toward Principles for the Design of Ontologies used for Knowledge Sharing". In formal Ontology in Conceptual Analysis and Knowledge Representation. Kluwer Academic Publishers. 1995.

[5] G. Guizzardi. "A methodological approach to development and reuse, based on formal domain ontologies" Master Degree. Federal University of Espírito Santo. 2000.

[6] T. Berners-Lee, O. Lassila, and J. Hendler. "The semantic web." Scientific American, 2001, pp. 5:34–5:43.

[7] N. Noy and D. Mcguinness,. "Ontology development 101: A guide to creating your first ontology,". [Online].Available:http://www.ksl.stanford.edu/people/dlm/papers/ontology101/ontology101-noy-mcguinness.html. [retrieved: 06, 2013]. 2001.

[8] F. Freitas. "Ontologies and the semantic web". Proceedings of XXIII Computer Sience Brazilian Society Symposium. Campinas: SBC. v. 8, 2003, pp. 1-52.

[9] N. Guarino, "Formal ontology and information systems," in Proceedings of FOIS98. Trento, Italia: IOS Press, pp. 3–15. 1998.

[10] I. Mirbel. "OFLOSSC, "An Ontology for Supporting Open Source Development Communities". In Proceedings of the International Conference on Enterprise Information Systems (ICEIS). 2009.

[11] W. Maalej and H. Happel. "A Lightweight Approach for Knowledge Sharing in Distributed Software Teams". In Proceedings of the Practical Aspects of Knowledge Management (PAKM). 2008.

[12] T. Dillon and G. Simmons. "Semantic Web support for Open-source Software Development". In Proceedings of the International Conference on Signal Image Technology and Internet Based Systems (SITIS). 2008.

[13] A. Chaves, I. Steinmacher, C. Lapasini, E. Huzita, and A. Biasão. "OntoDISEN: an Ontology to Support Global Software Development". CLEI Electronic Journal. 2011. v. 14, pp. 1-12.

[14] M. Fernandez, A. Gomez-Perez, and N. Juristo, "Methontology: from ontological art towards ontological engineering," in Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, Stanford, USA, 1997, pp.33–40.

[15] IEEE, "Standard for developing software life cycle processes". p. 96, may 1997, eEE Computing Society. Available: http://standards.ieee.org/catalog/olis/archse.html.[retrieved:06, 2013]. 1997.

[16] OWL. Web ontology language overview. Available: http://www.w3.org/TR/owl-features.[retrieved: 06, 2013]. 2009.

[17] J. Berners-Lee and O. Lassila, "The semantic web," Scientific American Magazine.[retrieved: 06, 2013]. 2001.

[18] Protégé. Protégé ontology editor. Online. [Online]. Available: http://protege.stanford.edu/doc/users.html. [retrieved: 06, 2013]. 2009.

[19] J2EE. JAVA Enterprise Edition. Available: http://oracle.com/technetwork/java/javaee/overview/index.html, [retrieved: 06, 2013]. 2013.

[20] Grails. Available: http://grails.org, [retrieved: 06, 2013]. 2013.

[21] Groovy. Available: http://groovy.codehaus.org, [retrieved: 06, 2013]. 2013.

[22] Google Web Toolkit. Available: http://gwtproject.org, [retrieved: 06, 2013]. 2013.

[23] Hibernate. Available: http://hibernate.org, [retrieved: 06, 2013]. 2013.

[24] J. Carroll, D. Reynolds, I. Dickinson, A. Seaborne, C. Dollin, and K. Wilkinson, "Jena: Implementing the semantic web recommendations" . pp. 74–83. 2004.

[25] SPARQL Query Language for RDF. Available: http://w3.org/TR/rdf-sparql-query, [retrieved: 06, 2013]. 2013.

[26] OWL API. Available: http://owlapi.sourceforge.net, [retrieved: 06, 2013]. 2013.

[27] P. Wongthongtham, E. Chang, and T. Dillon. "Ontology-based Multi-agent System to Multi-site Software Development". In Proceedings of the Workshop on Quantitative Techniques for Software Agile Process (QUTE-SWAP). (Newport Beach, USA). 2004.

[28] T. Dillon, G and Simmons, G. "Semantic Web support for Open-source Software Development". In Proceedings of the International Conference on Signal Image Technology and Internet Based Systems (SITIS). 2008.

[29] A. Ankolekar, K. Sycara, J. Herbsleb, and R. Kraut. Welty Chris. Internactional Conference on World Wide Web. Pg 575-584. 2006.

[30] A. Borges, R. Rocha, C. Costa, H. Tomaz, S. Soares, and S. Meira. "Ontologies Supporting the Distributed Software Development: a Systematic Mapping Study". In Proceedings of the International Conference on Evaluation & Assessment in Software Engineering (EASE). Porto de Galinhas, PE, Brasil. 2013.

[31] Files Repository of Mapping Study about Ontologies in Distributed Software Development: http://www.rgcrocha.com/ms, [retrieved: 06, 2013]. 2013.

Comparative Influence Evaluation of Middleware Features on Choreography DSL

Nebojša Taušan, Pasi Kuvaja, Jouni Markkula, Markku Oivo
Department of Information Processing Sciences
University of Oulu
Oulu, Finland
{nebojsa.tausan, pasi.kuvaja, jouni.markkula, markku.oivo}@oulu.fi

Jari Lehto
Department for Process Improvement
Nokia Siemens Networks
Espoo, Finland
jari.lehto@nsn.com

***Abstract*—Domain-Specific Languages for service interaction modeling in the embedded systems domain are generally considered insufficiently expressive. To fully represent what is relevant for the developers, service interactions are commonly modeled from two viewpoints: orchestration, which is the individual, and choreography, which is the global viewpoint. In the embedded systems domain, proposed modeling languages are focused on orchestrations, while choreography modeling is neglected. For this reason, we compared two middleware products, one from the automotive and the other from the telecom industry sector, and analyzed variations in the implementation of choreography relevant features. Our analysis shows the influences of implementation variations on language for choreography modeling. Our findings can be useful in developing a domain-specific language that will allow the full representation of choreographies in the embedded systems domain.**

Keywords-choreography; DSL;middleware; SOA; MDE

I. INTRODUCTION

Service-Oriented Architecture (SOA) is an architectural style that is commonly used in the development of large enterprise systems [1]. Recently, SOA has found its application in industrial sectors such as the automotive and telecom where it is used in the development of embedded systems [2] [3] [4]. This has opened an opportunity to transfer knowledge and technology from one domain to another, but also to extend existing knowledge and technology, so it can meet new challenges that are specific to the embedded systems domain.

Systems built based on the SOA style can be described as collections of autonomous applications, called services, which interact to fulfill the stakeholder's needs. Therefore, explicit representation of how services interact becomes an important aspect of SOA systems. According to Dijkman and Dumas [5] and Peltz [6], modeling of the service interaction aspect should comprise two viewpoints, orchestration and choreography. In short, orchestration shows service interactions from a single participant's point of view, while choreography shows a global, peer-to-peer interaction between participants. These viewpoints overlap in the sense that both illustrate how underlying services interact, but differ in the perspective, or in the viewpoint, from which they show the interaction aspect.

One approach to how service interaction aspect can be analyzed and specified is to use Domain-Specific Languages (DSL). DSLs, unlike general purpose languages are focused on one particular aspect or one particular domain of a software system. The main idea behind DSL usage is to shorten the development time, reduce errors, and improve the communication by enabling language support for concepts that are specific to the aspect of interest [7].

Modeling of service interaction aspects in the enterprise system domain is supported with several DSLs; examples are [8] [9] [10] [11] [12] for orchestration and [13] [14] [15] [16] for choreography. These languages, however, are not sufficiently expressive to represent interactions that may occur in embedded systems [17] [18] [19]. Therefore, new DSLs, or supplements to existing DSLs, have been developed for embedded systems.

In the telecom domain, Call Control eXtensible Markup Language (CCXML) [20] is used to controls the invocation order of telephony services. The drawback of CCXML is that it can invoke only services developed in telephony specific technologies. To overcome this limitation, a State Chart eXtensible Markup Language (SCXML) [21] was proposed. SCXML is a generic language for describing complex state machines. It complements CCXML by providing a generic state-machine framework and by enabling it to invoke services developed in telephony and non-telephony-specific technologies.

Vandikas and Niemoeller proposed SCALE [22] as a modeling language whose main goal is to enable modeling of telecom specific interactions, but also to allow convergence of telecom services and services developed in different domains and technologies. Similarly, SPATEL [23] language targets the same problem, and offers technology-independent primitives that can be used for the development of telecom services where large numbers of resources needs to interact over different protocols.

The described DSLs enable service interaction modeling; however, they support only the modeling of individual participant point of view, or orchestration. Global view on interactions, or choreography, is not natively supported with their language entities.

Service interactions can be modeled with domain-agnostic languages or by modifying languages from different domains. A case in the automotive domain is reported by Fiadeiro et al. [24] where SENSORIA

Reference Modeling Language (SRML) is used. SRML [25] is designed to be a domain-agnostic language, with strong expressiveness for SOA, and to be easy for formalization.

Business Process Execution Language (BPEL), which is used for modeling enterprise service interactions, is modified by Iwai et al. [18] to represent the complex interactions of services in automotive domain. As in the case of telecom DSLs, these approaches target only the orchestration point of view. With the exception of the work by Tsai et al. [26], the current state-of-the-art in service interaction modeling led us to conclude that less focus is put on modeling choreography aspects in service-oriented embedded systems development.

To bridge this gap, part of the work done during the AMALTHEA [27] research project was to develop a DSL suitable for choreography modeling in the embedded systems domain. The language is one of the several tools in the tool-chain platform that is implemented within the project. During implementation tasks, we adopted the guidance for DSL development proposed by Merink et al. [28]. This guidance is organized according to DSL development phases, and in this article, we will present our findings from the analysis phase. During this phase, together with industry partners, we analyzed middleware products that are used in the automotive and telecom industry. There are two main reasons why middleware analysis is relevant for the development of DSL.

The first reason originates from the DSL development guidance [28], according to which the input to the DSL analysis phase can be technical documentation, knowledge provided by experts, customer surveys, and the existing source code base. Accordingly, for our analysis, we used expert's knowledge, and technical documentation of two middleware products. Middleware, and its documentation, is an unavoidable part of any large software system, and its main responsibility is to enable seamless interaction between system parts [29]. Accordingly, it is a valuable source of service interaction-related knowledge, which is the key result of the DSL analysis phase.

The second reason is related to Model-Driven Engineering (MDE) [30], which is the engineering approach in companies that participated in this project. In the MDE approach, relevant system aspects are modeled using DSLs. Unlike in traditional, document-driven approaches, the developed models in MDE are executable or readable by tools. This allows automatic analysis, transformation from one system representation (one model) to another, and automatic test and source code generation. Source code generated from different models relies on, or executes on top of, middleware. Middleware, however, imposes rules and constraints to that code that must be understood and followed during modeling [31]. One way to enable this is to include and enforce those rules and constraints with DSLs. This way, DSLs and their models become tightly coupled to the middleware on top of which the developed application will execute.

Middleware products support developers by providing them with features that hide complex low-level tasks [29]. Different middleware products, however, implement features differently, which introduces variations in implementation and in extent of support the feature provides. If features, with the rules and constraints they impose, are to be addressed with DSL, these variations must be taken into account. To better understand the relationship between variations and DSL development, in this study, we will answer the following research question.

How do variations in the implementation of middleware features influence the implementation of the DSL for choreography modeling?

Answering this research question will help choreography DSL developers by pointing out which language entities are influenced by feature implementation variation and how. To answer this research question, we identified choreography-relevant features and their implementation variants (Section II). Based on these features, we compared two middleware products, identified influenced choreography language entities, and described the influence in more detail (Section III). Following is the discussion on benefits that can be expected from DSL that includes implementation variations (Section IV). Finally, we summarize the study findings and describe the future work (Section V).

II. Research Design

Analysis phase of DSL development is conducted by adopting DESMET [32] approach for evaluation, and Goal Question Metrics (GQM) method [33] for feature and scale derivation. DESMET proposes nine methodological approaches for evaluating methods, tools, and technologies [34], and defines the criteria based on which an evaluator can select the most appropriate one. Based on the evaluation context, nature of the impact, nature of the evaluation object, and maturity of the item criteria, we have selected *feature analysis in screening mode* (FA) approach for this study. The evaluation context criterion recommends FA in cases where the object under evaluation will be sold as a part of a larger product. Middleware, as the object under evaluation, is a part of the overall system that resides between the operating system and application. The nature of the impact criterion recommends FA in cases when a study produces qualitative results. This is in line with this study, since we are aiming to show the influence of implementation variations on DSL development. The nature of evaluation object criterion recommends FA when tools are in the focus of evaluation. Middleware is primarily a technology, but it can also be approached as a tool for supporting a developer's work. The maturity of the item criterion proposes FA when large amounts of information about study object are available. This corresponds with the middleware products evaluated in this study. The first is the de facto standard in the automotive industry. The second is a proprietary technology owned by the company that participates in this research project.

A. Analysis Procedure

DESMET FA is a qualitative approach to evaluation. It formulates features according to what users expect from the method, tool, or technology, and derives corresponding scales that measure the extent to which the candidate method, tool, or technology conforms to the formulated features. When FA is done in the screening mode, feature derivation and evaluation is done by a single person based on public documentation only. Accordingly, during this study, middleware features that are seen as relevant for choreography DSL are identified, their scales are derived, and, based on those, two middleware products are evaluated. Contrary, instead of one, four researchers and one industry expert collaborated during feature derivation and evaluation. Research collaboration consisted of face-to-face meetings, teleconference meetings, and exchange of email messages.

DESMET FA evaluation consists of six steps, which we followed during this study, and described in the text below.

Step 1: Identify the candidate method/tool/technology. This research project, brought together researchers and experts from automotive and telecom industry. In both industries, different middleware products are used for systems development and for this analysis, AUTOSAR [35] and LISA were chosen. The reason for choosing AUTOSAR, over other products such as OSGi, is that it represents a de facto standard in automotive industry. It is a result of a global partnership of automotive manufacturers and suppliers, which aims to become the standardized architecture for automotive software. AUTOSAR is also a dominant middleware in automotive companies which participated in this research project. LISA stands for Light Intelligent Software Architecture and it is a proprietary middleware solution for the development of telecom systems. The reason for choosing LISA for this analysis is that it is still a prototype and open for modifications. This motivated telecom experts to compare LISA against more mature AUTOSAR and to learn about similarities and differences in the implementation of two products.

Regardless of differences in many aspects of automotive and telecom systems, closer inspection of the middleware products revealed a number of similarities. These similarities form a basis for comparing AUTOSAR and LISA. Figure 1, illustrates the similarities between the two systems, and shows the position of middleware within them. With reference to Figure 1, these similarities are: a) Systems consist of heterogeneous hardware devices (Hardware A, B, and C). b) Hardware devices are interconnected with heterogeneous network technologies (labeled with 1 and 2). c) Hardware devices can have different operating systems (OS 1, 2, and 3). d) The middleware homogenizes hardware devices, network and operating systems. e) The middleware hides hardware, network, and OS complexities by offering higher level application programing interface (API) to

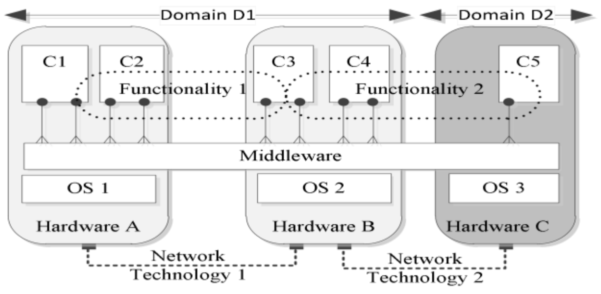


Figure 1. Architectural similarities of two systems

application components. f) Application components (C1–5) reside in hardware devices and run on top of middleware. g) Applications are realized with one or more application components. h) Hardware and application components may or may not be under the control of a single authority (Hardware A and B belong to D1 domain, Hardware C belongs to D2 domain, while domain here denotes different organization units or different companies). i) End-user perceived functionality (Functionality 1 and 2) is realized through application component interactions. j) Application components that realize functionalities can reside on the same or on different hardware devices. Described similarities are the key argument why we consider AUTOSAR and LISA comparable and therefore they will be explained in more detail.

Step 2: Devise the assessment criteria. FA is a comprehensive approach to evaluation. Besides technical issues, the method proposes to evaluate features from economic, cultural, and different quality aspects such as maintainability or portability. To narrow down the scope of evaluation, we applied the GQM method during the derivation of features and corresponding judgment scales. The importance of the clear goal definition is highly stressed in the GQM approach since it provides a converging point for future scales and it reduces the number of possible measurements [36]. It is important to note the misalignment in terminology within DESMET and GQM. In DESMET, judgment scales are used to estimate the derived features, while in GQM, scales are used to estimate the measurements. Therefore, in this study, the terms measurements and features can be considered equivalent since both are used for answering questions formulated according to a specified goal. Goal specification is further facilitated with a GQM template [33], which consists of five key-value tuples. A study goal, based on this template, is presented in Table I.

The first tuple in the template defines the object under investigation. In this study, the object under investigation is the middleware. The second tuple defines the purpose for analyzing that object. In this article, the purpose is to learn which middleware features should be considered during the development of choreography language. Accordingly, the choreography aspects of service interaction support in middleware are the quality focus against which we analyzed LISA and AUTOSAR. Viewpoint narrows the scope of learning by focusing it on a specific role in the development process. We selected the software architect role because it is responsible for middleware-related decisions, and because LISA and AUTOSAR can be easily compared in architectural terms such as components, services, interface, and message. Lastly, this international research project is

TABLE I. FEATURE ANALYSIS GOAL BASED ON GQM TEMPLATE

Key	Value
Analyze the :	Middleware
For the purpose of:	Learning
With respect to (quality focus) :	Service interaction aspects relevant for choreography modeling
From the viewpoint of:	Software architect
In the context of:	Research project

the context in which the evaluation took place.

Specified in this way, the goal guided our collaboration with the industry experts and our study of the literature. This resulted with the definition of three questions whose answers will contribute to the goal accomplishment. Questions are broken into features relevant for the choreography DSL, and for each feature, a corresponding ordinal scale is derived. Here, we will emphasize that scales are derived based on the extent of support the feature provides to developers. A higher feature score corresponds to higher flexibility, less effort, and less cognitive burden for developers. Scales do not measure the variations in technology that is used for feature implementation.

The first question, based on the defined goal, is: How does middleware support the invocation of services offered by different systems or system parts? To answer this, three features are identified and explained in the following text. *Functionality access* is the first. It concerns middleware support for invoking services that use different interfacing technologies, e.g., Web Service Description Language (WSDL) and Interface Definition Language (IDL). *Location transparency* is the second. It concerns middleware support for binding service requesters and providers. Location Transparency can be realized using requester's criteria based on which middleware selects the provider, logical names based on which physical location of the provider is resolved, or by plain routing. *State information* is third. It concerns types of state information that middleware monitors. State information types are classified into service, session, and functions categories. Service indicates the state of the application or component that implements the service. Session indicates the state of the interaction between two services. Function indicates the state of the composition of services that fulfills a system-level task. Table II shows the extent of support the middleware provides for identified features.

The second question based on the defined goal is: How does a middleware product supports issues related to messages? Message is used in a broad meaning, and covers both the format of the message and the type of data that is carried. To answer this question, three features are identified and explained. First is *message format*. It indicates which message and data formats can be processed. Message format examples can be Session Initiation Protocol (SIP) or Simple Object Access Protocol (SOAP), while the data format can range from streams of bits to documents written in plain text or in eXtensible Markup Language (XML). Data format is commonly defined by the message format that carries it. Second is *message transformation*. It concerns middleware support for transformation of messages from one format to another. Third is *interaction scenario*. It shows the middleware ability for processing predefined ordering of message exchange occurrences. These features and extent of support are given in Table III.

TABLE II. FEATURES FOR SERVICE INVOCATION SUPPORT

Feature	Scale	Scale description (implementation variants)
Functionality access	2	Middleware supports standardized interfacing technology specific for an industry sector
	1	Middleware supports key interface technologies
	0	Middleware imposes single interface technology
Location transparency	2	Middleware selects service provider, resolves its location and routs the request
	1	Middleware resolves service provider's location and routes the request
	0	Request contains details that are necessary for binding (provider name, physical location, etc.). Middleware only routs request to provider
State information	2	Middleware provides state information on function, session, and service level
	1	Middleware provides state information on session, and service level
	0	Middleware provides state information on service level

TABLE III. FEATURES FOR MESSAGING SUPPORT

Feature	Scale	Scale description (implementation variants)
Message format	2	Middleware processes message format that is standardized within an industry sector
	1	Middleware processes key message formats that are used in an industry sector
	0	Middleware imposes message format
Message transformation	2	Middleware transforms key message formats, and allows developers to create custom pluggable transformation additions
	1	Middleware transforms key message formats; middleware vendors supply additional transformations through product updates.
	0	Middleware does not provide message transformation services
Interaction scenario	2	Middleware processes custom definitions of message interaction scenarios
	1	Middleware service supports generic interaction scenarios
	0	Middleware does not support the processing of interaction scenarios

Lastly, a third question based on the defined goal is: How is a message transmitted from its origin to its destination? Two features are identified and explained. *Protocol support* is the first, and it shows middleware support for a variety of communication protocols. As is the case with messages, the term protocol here is used to cover all types of protocols, ranging from lower-level network specific protocols, such as Controller Area Network (CAN), to high level application protocols, such as Hypertext Transfer Protocol (HTTP). *Protocol translation* is the second identified feature, and it shows how middleware supports the translation of one protocol to another. These features and implementation variants are shown in Table IV.

Step 3: Compiling information about the study object. To evaluate the candidate technologies, relevant

TABLE IV. FEATURES FOR MESSAGE TRANSMISSION SUPPORT

Feature	Scale	Scale description (implementation variants)
Protocol support	2	Middleware supports different protocols by providing protocol-independent communication service
	1	Middleware supports different protocols by providing protocol-dependent services
	0	Middleware imposes the protocol
Protocol translation	2	Middleware communication services hide protocol translations from services
	1	Middleware provides distinct services for protocol translation
	0	Middleware does not provide translation support. Services are responsible for translation

documentation needs to be collected and studied. This research project provided a context that allowed us to collect high-quality, company-specific documents, and to capture the knowledge of company experts in meeting notes, email discussions, and workshop summaries.

Step 4: Scoring of features. Based on the gathered information, middleware products are evaluated against derived features. The process of scoring consisted of the initial score proposals and discussion. During score proposal, each team member proposed a score for each feature. During the discussion, the differences in score proposals are aligned.

Step 5: Analysis of the score. To decide which method, tool, or technology best fits the needs of the most target users, feature scores are analyzed. The goal of this evaluation, however, is not to select between middleware products. Our goal is to learn about middleware features and to show how their variants can have an influence on the language for choreography modeling. For this purpose, we used the meta-model for choreography language defined in [37]. This model defines what is necessary for the development of global interactions, and represents a foundation for the development of choreography modeling languages. It consists of attributes enclosed in entities that are interconnected and grouped into model subsets. We used this model to identify how, and which of its entities are influenced by the variations in middleware's feature implementation.

Step 6: Presenting a report on the evaluation. The research findings are summarized in a technical report.

III. RESEARCH FINDINGS

The research findings are divided into two groups. The first group consists of feature scores and the rationale behind scoring. In the second, we explain how variations in feature implementation have an influence on DSL for choreography modeling.

A. Features Scores

The rationale behind scoring is based on an in-depth analysis of the technological solutions and concepts that are used for feature implementation and on industry expert's evaluation of the extent of support the feature provides. The implementation details used in AUTOSAR and LISA for an identified feature are described below.

Functionality access: To describe what a service can provide, what other services it uses, and how to invoke the service, AUTOSAR developed the AUTOSAR Interface [38]. This interface has a formal structure that describes all aspects required for the invocation of functionality. LISA, in contrast, has no structured description of a service. LISA facilitates access to service functionalities by offering a proprietary API through which applications (services) register and publish their functionalities. Through this, potential clients are able to invoke the functionalities they need. Other than function names, no additional details are provided.

Location transparency: Both middleware products studied in this evaluation provide support for binding by hiding the location details of services. A service can invoke another services' functionality using only its logical names, while the middleware pairs logical names with the services' functionality and its physical location. This allows services to be moved to different hardware devices, and if there is a need, to change its implementation details. Since the functionality is invoked using logical names, flow of service interactions is not affected.

State information: Both AUTOSAR and LISA provide state information on the service and session levels. On the service level, AUTOSAR monitors the state of the runnable concept [38], while LISA allows for monitoring of each service that implements the proprietary LISA-specific addresses. On the session level, AUTOSAR's inter-runnable communication state information is provided with global variables and/or shared memory monitoring [38], while LISA provides session-level information by monitoring its implementation of message queues.

Message format: AUTOSAR services exchange information using three standardized variable groups: data element, mode declaration, and application error [38]. The data element is the piece of information transmitted between services. This information is sent to, and received from, the service's operations, and it can be any primitive type, such as integer or float, or a collection of primitive types referred to as the complex type. Mode declarations define data for the service mode configuration, while application errors carry the information about error occurrences within a service or during communication. In AUTOSAR, variables are exchanged by passing them to functions directly, and no additional messaging technology is used.

Messages in LISA are exchanged using proprietary messaging technology. Before a message of any type is sent, it is wrapped up in a LISA-specific message format and routed to the destination. On arriving at the destination, it is unwrapped and parsed by the receiver.

Message transformation: In AUTOSAR, the object of transformation is the data element variable group, and this task is appointed to the runtime environment (RTE). Transformation definitions are provided by developers, and, based on them, RTE can perform several types of transformations. Examples are transformations to/from different linear-scaled data representations, different text-table data representations, and transformation of composite data representations [39]. LISA does not provide any features for transforming messages. Instead, it is the responsibility of the sending or receiving service to preprocess the message so that it can be used by the receiving application.

Interaction scenarios: There are two generic scenarios that describe a message exchange in AUTOSAR, Client-Server and Sender-Receiver [39]. Client-server involves the client, who requires the functionality and server that provides that functionality. The client initiates the communication by requesting the server to perform the functionality and if necessary it provides one or more parameters. The server performs the required function, and dispatches a response to the client. Invoking a function is performed by RTE, and these invocations can be either asynchronous or synchronous. The sender-receiver involves the sender of the message and one or more receivers. This is one way, the asynchronous interaction scenario, and any reply sent by receiver is seen as a separate sender–receiver communication. The same scenarios exist in LISA. The difference is that in the case of AUTOSAR these patterns are explicitly defined within the interface, while in LISA, no such definition exists. The client-server scenario occurs when one service invokes the operation of other service, while that of the sender-receiver is realized through multicast message delivery. No custom definition of interaction scenarios is possible in either product.

Protocol Support: Both products under evaluation provide unique services that hide the transport protocol and networking technologies and allow the inclusion of additional ones without modifications at the application level. In AUTOSAR, this is realized with a group of modules called communication services, and an Interface-Protocol Data Unit (I-PDU protocol) [40]. These concepts provide an interface to the communication network, API for network management and diagnostics, and hide protocol and network-level details from applications. Similarly, LISA has developed a proprietary module called the Media Module. This module abstracts different protocols and network types, such as Ethernet, Socket, and W-LAN, and enables uniform transmission of messages over a heterogeneous network environment.

Protocol translation: AUTOSAR and LISA provide middleware-specific, communication protocols to services, and, during message exchange, this is the only protocol services are aware of. Internally, middleware translates this, into a protocol specific to, e.g., a physical network through which the message is transported. In the case of AUTOSAR, the Communication Services pack and unpack messages to and from the I-PDU, which are then passed to network specific modules for transmission over the physical network. Likewise, LISA uses Media Module and its protocol at communication endpoints, but translates it to the network specific protocols used during transmission.

Based on analysis, extent of support the feature provides to developers is evaluated and summarized in Table V.

B. *Language Entities Influenced by Variants*

To understand the influence of variations in feature implementation on DSL for choreography modeling, we studied a meta-model proposed in [37]. This resulted in the identification of language entities whose implementation varies depending on the extent of the support feature provides to developers. To express variations in language entity implementation, we used language constructs such as sub-entity, attribute, and relationship multiplicity. Identified entities are as follows:

Participant: an entity that represents any logical encirclement within the system that has a degree of autonomy, and provides functionality for other Participants in the system. An example can be an accounting unit within an enterprise, a braking subsystem in the car, or a home subscriber server in telecom network. From implementation point of view, a Participant can encompass a component, collection of components, or an entire application.

To access a Participant's functionality different interfacing technologies are offered and these should be supported by middleware so that Participants can seamlessly interact. In Table II, we proposed implementation variants for a functionality access feature that can influence how a Participant, as a language entity, is implemented.

In the case of AUTOSAR, due to the use of unique and standardized interface across industry sector, Participant entity should define the attributes that are needed to describe the AUTOSAR interface only. In LISA, no structured description for accessing functionality is defined. IN this case, a Participant should include attributes that describe proprietary, LISA-specific invocation methods. In the case that a middleware product supports different interfacing technologies, a Participant entity should implement distinct sub-entity types with attributes specific to each of the supported technologies. The relationship between the Paritcipant and sub-entity should be constrained to a one-to-one relationship.

Implementation of a Participant entity is also dependent on the location transparency feature. In Table II, we proposed variations for this feature, which we see as influential for an entity implementation. Since AUTOSAR and LISA use logical names for accessing the service functionality, in both cases, a participant should provide attributes where these names will be recorded.

Role: an entity that represents the responsibility of the Participant in the scenario, and as a choreography language entity, it is a part of the participant. One Participant can have different Roles in different interaction scenarios. An example can be a Role of the organization unit that participates in choreography as "buyer" in one and "seller" in another scenario or a Role of the car engine control, which can be a "manager" in one, and a "data provider" in another scenario.

From an implementation point of view, a Role can be identified with one or more functionalities offered by Participant. Therefore, a Role must implement sub-entities for describing each of the functionalities that are included in it. Since a Participant can use different interface technologies, a set of dedicated sub-entity types should be defined, where each type would specify attributes for describing functionalities according to each of the supported technologies. In case of AUTOSAR, a Role entity should consist of functional descriptions defined according to the AUTOSAR interface. In the case of LISA, a Role should describe the functionalities based on LISA's proprietary technology for accessing applications.

TABLE V MIDDLEWARE EVALUATION RESULTS

Question	Feature	AUTOSAR	LISA
Invocation support	Functionality access	2	0
	Location transparency	1	1
	State information	1	1
Message support	Message format	2	1
	Message transformation	2	0
	Interaction scenario	1	0
Message transmission	Protocol support	2	2
	Protocol translation	2	2

Interaction: an entity that represents the exchange of information between two Roles. Exchange of information here is used to denote the ordering of one or more message exchange occurrences that together realize the Interaction. Call control application, for example, can have a Role of a service provider and must interact with the verifier, which is the Role of the subscriber information repository, to verify that a certain subscriber can use the call service. This Interaction can be realized with two message exchanges that occur in a predefined order. First, a provider sends the message with subscriber info to the verifier. Second, the verifier processes the message and sends the response back to the provider.

From an implementation point of view, an Interaction describes the order of message exchange occurrences between Roles. When implemented in language, Interaction is expressed in terms of generic (or predefined) message exchange scenarios. The idea behind this is that all message exchange scenarios conform to a single, or a combination, of generic exchange patterns. Therefore, Interaction entity implementation depends on which patterns are identified and used within an industry sector, and how they are supported by middleware product. In Table III, variants for Interaction scenarios are proposed.

AUTOSAR communication services recognize two generic scenarios, client-server and sender-receiver. Here, the Interaction entity should provide attributes for recording the two identified patterns. LISA offers no support for generic scenarios, an entity here can be implemented to allow unstructured textual description of message exchange ordering. These descriptions can be used to facilitate communication and analysis tasks.

Interaction entity implementation depends also on the implementation of message translation and protocol translation features. Participants engaged in interaction may require the translation of message content since the format in which the information is sent, is not always the format that the receiving Participant can process. An example can be a Participant that sends a SOAP message to the Participant that can receive only SIP messages. Middleware can provide features for message transformation, and implementation variants of this feature are proposed in Table III. Depending on how middleware implements the feature, Interaction will need to adopt accordingly.

AUTOSAR allows developers to define message transformations. The language entity, in this case, should include attributes for linking entities with defined transformations. LISA offers no such facilities. Including transformation-related data in an Interaction entity can only be used for documenting purposes.

Similarly to message transformation, Participants can use different communication protocols for message transmission. How middleware implements the protocol translation feature also influences implementation of the Interaction entity, and in Table IV, implementation variants are proposed.

AUTOSAR and LISA provide a feature for protocol translation, and in both cases, translation is hidden from (or transparent to) Participants that are interacting. This is accomplished by the translation feature which is a part of uniform communication service that is offered by both middleware products, and used by the Participant for communication. The Interaction entity therefore doesn’t need to include attributes for describing translations of the protocols.

In cases when middleware doesn’t support protocol translation, this task should be implemented by applications that realize the Participants. In cases when middleware implements distinct translation services for each protocol, the Interaction entity should include attributes for recording the details necessary for linking the entity with translation services.

Message Content Type: A message carries the information that is exchanged between the Roles. The format of those messages can be different, and each format specifies the types of data it can carry. Thus, the purpose of this entity is to describe those message formats.

This entity is part of the Interaction. How it is implemented in language, depends on the message formats it must be able to describe. For this reason, in Table III, we proposed implementation variants for message format support. In AUTOSAR, messages are standardized, and to define them, an entity should include only attributes relevant for the definition of AUTOSAR messages. In LISA, different message formats are supported. Still, due to the wrapping technology it uses, for entity implementation, only attributes for wrapper description should be included.

State Variable: Roles engaged in interaction can have different states based on the information that is exchanged. The value of this entity is predefined, and its purpose is to hold those values. An example of a State Variable can be “Verification State”. Based on interaction condition, a variable can hold one of two predefined values, “verification sent” or “send error”.

As a language entity, the State Variable entity is a part of the Role, and its implementation depends on state information provided by the middleware product. In Table II, we proposed implementation variants for the State Information feature. These variants express different types of state variables and influence the implementation of a language entity. Both AUTOSAR and LISA provide state information that is relevant for service- and session-level state descriptions. The language entity should, therefore, provide attributes with predefined values for capturing those items of information.

Channel Variable: Its main purpose is to store the information that is necessary for sending the message. Part of this information is, for example, the protocol that defines the rules that must be followed during message transmission. Since participants involved in interaction can use different protocols, middleware products should support them, if seamless message exchange is to be achieved.

As a language entity, the Channel Variable entity is part of an Interaction entity, and the protocol-related information that it will include depends on variations in protocol support of the middleware product. In Table IV, we proposed implementation variants that are derived based on the amount of protocol information middleware requires from

TABLE VI. EVALUATION SYNTHESIS SUMMARY

Identified Entities	**Identified Features**	**Scales**	**Influence of Feature Implementation Variations on Language Entities**		
			Sub-Entity	**Attributes**	**Relationship**
Participant	Functionality access	2	No influence	Describing standardized interface technology	No influence
		1	Distinct Type of Sub-Entity per supported interface technology	Distinct attribute set per Sub-Entity type for describing supported interface technology.	One Participant can have one interface technology
		0	No influence	Describing imposed interface technology	No influence
	Location transparency	2	No influence	Describing criteria for service selection	No influence
		1	No influence	Data for resolving service invocation	No influence
		0	No influence	Data for routing service request to provider	No influence
Role	Functionality access	2	Sub-Entity per functionality that is included in Role	Describing functionality according to standardized interface technology	One Role can have one or more functionalities
		1	Sub-Entity per functionality that is included in Role	Describing functionality according to Participant's interface technology	One Role can have one or more functionalities
		0	No influence	Describing functionality according to imposed interface technology	No influence
Interaction	Interaction Scenario	2	Sub-Entity for custom interaction scenario	Description of custom interaction scenario	One Interaction can have one interaction scenario
		1	No influence	Attribute and predefined values for describing supported scenario	One Interaction can have one interaction scenario
		0	No influence	No influence	No influence
	Message transformation	2	No influence	Attributes for relating Interaction with transformations elements in middleware	No influence
		1	No influence	Attributes for relating Interaction with transformations elements in middleware	No influence
		0	No influence	No influence	No influence
	Protocol translation	2	No influence	No influence	No influence
		1	No influence	Attributes for relating Interaction with translation elements in middleware	No influence
		0	No influence	No influence	No influence
Msg. Content Type	Message format	2	No influence	Describing standardized message format	No influence
		1	Distinct Type of Sub-Entity per supported msg. format	Distinct attribute set per Sub-Entity type for describing supported msg. formats	Msg. Content Type have one msg. format
		0	No influence	Description of imposed message format	No influence
State Variable	State information	2	No influence	Attributes and predefined values on functional, session and service level	No influence
		1	No influence	Attributes and predefined values on session and service level	No influence
		0		Attributes and predefined values on service level	
Channel Variable	Protocol	2	No influence	No influence	No influence
		1	Distinct Type of Sub-Entity per supported protocol	Distinct attribute set per Sub-Entity type for describing protocol dependent communication. services	
		0	No influence	No influence	No influence

the Channel Variable. In both AUTOSAR and LISA's case, protocol details are hidden from (or transparent to) the Channel Variable by providing uniform communication services. To transmit a message, only functionality name and message are required, while the protocol details are handled by the middleware service.

In Table VI, we summarized how variations in feature implementation influence on the implementation of the identified choreography language entities. Depending on the extent of support the middleware feature provides to developers, language entity will implement different combination of sub-entities, attributes, and relationship multiplicity.

IV. DISCUSSION

Implementation variations of identified middleware features can influence the implementation (or supplementation) of a DSL for choreography modeling. Accordingly, variations represent a valuable source of information that needs to be considered during DSL development. There are several reasons why the inclusion of this information in language can be beneficial from the software development point of view. The most important reasons are described in the following text.

Broadening the scope of DSL in the development process: Choreography DSL is an analytical tool that specifies the contractual agreement between different sub-systems. By including middleware-specific data into DSL, besides being analytical artifacts, specified models become implementation artifacts as well. A model's role in implementation is best visible in the MDE approach, where a chain of model-to-model transformation events aims to end with generated source code. To facilitate seamless transformations, and be in compliance with middleware-induced assumptions, choreography DSL should include middleware-specific information as well.

Facilitation of communication: DSL for choreography modeling offers concepts and semantics that are needed for system analysts to agree on global service interactions. When DSL includes middleware-related information, completing models requires additional, technical-related, expertise. This way choreography modeling pulls together experts from different development areas who are cooperating on the same model and communicating using concepts and semantics that are imposed by the DSL.

Easier introduction of new developers: To develop applications on top of middleware, developers must learn and follow middleware-induced assumptions. This represents a cognitive burden for new developers, and makes modeling error-prone. When middleware concepts, rules, and constraints are built in DSL, following them comes naturally since the language itself guides the work with concepts and prevents the developer from breaking the middleware imposed rules and constraints.

A. Validity threats

As an approach, the FA-Screening mode has medium costs in time and resources, but carries a high risk for the confidence in findings. This is understandable, since the entire evaluation represents the subjective stance of a single evaluator, based only on public documentation analysis. To decrease the risk, several measures were applied during the research design. The first measure is related to the number of evaluators. Instead of one, our analysis procedure included five evaluators. Joint work ensured that the findings are based, not only on a single person's stance, but encompass the opinions of five persons with different backgrounds and expertise. The second measure is related to sources of data. Instead of consulting only publically available documents such as standards or vendor material, in *Step 3* we used company-specific material and an industry expert's knowledge.

The authors of this article believe that applying these measures during study design increased the study objectivity, and decreased the confidence risk related to findings. Additionally, researchers worked under NDAs to assure the confidentiality of company documentation, and the industry expert was familiar with the issues being researched and the way company-specific data will be treated. According to Miles and Huberman [41], described measures and practices should reduce the validity threats.

Additional drawback is that, during the score analysis step, we used the meta-model that assumes the usage of Web-Services. Web Services are only one of several component technologies that can be used for telecom and automotive systems development. Still, the model leaves enough space for customization, and therefore we found it to be generic enough for discussing choreographies in the context of other component technologies as well.

V. CONCLUSION AND FUTURE WORK

The application of SOA in an embedded systems domain appears to continue to grow, and with it the need to model service interaction aspects is increasing. Using DSLs for modeling different system aspects has proven to be a good practice, and, with the growing adoption of MDE, their significance in development process will continue to grow. The research presented in this article supports this trend by focusing on the relations between the development of a DSL for choreography modeling and the underlying middleware.

Our findings suggest that the implementations of identified choreography language entities can differ depending on how middleware features are implemented. In Table VI, we describe an explanation of how this can be done. The same table can be used to answer the research question stated at the beginning. In short, based on feature implementation variations, identified language entities, which are Participant, Role, Interaction, Message Content Type, State and Channel Variable, will be implemented using different combinations of language constructs such as sub-entities, attributes, relationships, and value constraints. Concrete instances of sub-entities, attributes, and values are specific to industry sector, underlying middleware, and feature implementation technology and therefore not discussed in this article.

Future Work: The derived list of middleware features is certainly not complete. Additional features that are relevant for choreography modeling can be proposed, for example the feature for security issues. Furthermore, middleware analysis is not sufficient for DSL specification. Other problems and solution space artifacts should be analyzed to provide the needed expressiveness of the DSL. Lastly, a case study to collect broader opinions and suggestions from industry experts regarding Choreography DSL should be conducted. Future work will, therefore, continue in the above mentioned directions.

ACKNOWLEDGEMENT

This research has been supported by ITEA2 and TEKES. The authors are grateful to AMALTHEA partners for their cooperation and to Sanja Aaramaa, Markus Kelanti, and Jarkko Hyysalo for their comments and suggestions.

REFERENCES

[1] D. Krafzig, K. Banke, and D. Slama, Enterprise SOA: Service-Oriented Architecture Best Practices. Prentice Hall Profesionall, 2005.

[2] A. Scholz, et al. "∈ SOA-Service Oriented Architectures adapted for embedded networks," 7th IEEE International Conference on Industrial Informatics, IEEE, June 2009, pp. 599-605.

[3] L. Bocchi , J. L. Fiadeiro, and A. Lopes, "Service-oriented modelling of automotive systems," 32nd Annual IEEE International Computer Software and Applications Conference, IEEE, July 2008, pp. 1059-1064.

[4] T. Blum, N. Dutkowski, and S. Magedanz, "Evolution of SOA concepts in telecommunications," Computer, vol. 40, no. 11, Nov. 2007, pp. 46-50.

[5] R. Dijkman and M. Dumas, "Service-oriented design: A multi-viewpoint approach," International journal of cooperative information systems, vol. 13, no. 4, Dec. 2004,

pp. 337-368.

[6] C. Peltz, “Web services orchestration and choreography,” Computer, vol. 36, no. 10, Oct. 2003, pp. 46-52.

[7] M. Fowler, Domain-specific languages. Addison-Wesley Professional, 2010.

[8] F. Leymann, “Web Services Flow Language (wsfl 1.0),” IBM Software Group, May 2001, [Online]. Available: http://cin.ufpe.br/~redis/intranet/bibliography/standards/leymann-wsfl01.pdf, [Retrieved: Jun, 2013]

[9] WfMC, “Process Definition Interface - XML Process Definition Language,” Ver. 2.2, WfMC Standard, Doc. Number WfMC-TC-1025, Aug. 2012.

[10] OASIS, “Web Services Business Process Execution Language Ver. 2.0,” OASIS Specification Draft, Aug. 2006

[11] OASIS “ebXML Business Process Specification Schema Technical Specification v2.0.4,” OASIS Standard, Dec. 2006

[12] OMG, “Business Process Model and Notation”, Ver. 2, Object Management Group specification, Jan. 2011.

[13] J. Zaha, A. Barros, M. Dumas, and A. T. Hofstede, “Let’s dance: A language for service behavior modeling,” On the Move to Meaningful Internet Systems: CoopIS, DOA, GADA, and ODBASESE, Springer, Nov. 2006, pp. 145-162.

[14] W3C “Web services choreography description language Ver. 1.0,” W3C candidate recommendation, Nov. 2005.

[15] G. Decker, O. Kopp, F. Leymann, and M. Weske, “BPEL4Chor: Extending BPEL for Modeling Choreographies,” IEEE International Conference on Web Service, IEEE, July 2007, pp. 296-303 .

[16] A. Barker, C.D. Walton, and D. Robertson, “Choreographing web services,” IEEE Transactions on Services Computing, vol. 2, no. 2, June 2009, pp. 152-166.

[17] G. Bond, E. Cheung, and I. Fikouras, “Unified telecom and web services composition: problem definition and future directions,” Proceedings of the 3rd International Conference on Principles, Systems and Applications of IP Telecommunications, ACM, July 2009, pp. 1-12.

[18] A. Iwai, N. Oohashi, and S. Kelly, “Experiences with automotive service modeling,” Proceedings of the 10th Workshop on Domain-Specific Modeling, ACM, Oct. 2010, pp. 1-6.

[19] L. Lin and P. Lin, “Orchestration in Web Services and real-time communications,” IEEE Communications Magazine, vol. 45, no. 7, July 2007, pp. 44-50.

[20] W3C, “Voice Browser Call Control: CCXML Version 1.0,” W3C recommendation, July 2011.

[21] W3C, “State Chart XML (SCXML): State machine notation for control abstraction,” W3C working draft, Aug. 2013.

[22] K. Vandikas and J. Niemoeller, “SCALE - A language for dynamic composition of heterogeneous services,” Ericsson AB, Nov. 2010, [Online]. Available: http://www1.ericsson.com/res/thecompany/docs/journal_conference_papers/service_layer/101215_scale.pdf, [Retrieved: Jun, 2013]

[23] A. J. Paulo, A. Baravaglio, M. Belaunde, P. Falcarin, and E. Kovacs, “Service Creation in the SPICE Service Platform,” Proceedings of the 17th Wireless World Research Forum Meeting, Heidelberg: Wireless World Research Forum, Nov. 2006, pp 1-7.

[24] J. Fiadeiro , A. Lopes, and L. Bocchi, “A formal approach to service component architecture,” Web Services and Formal Methods, Springer, Sept. 2006, pp. 193-213.

[25] J. Fiadeiro, A.Lopes, L.Bocchi, and J.Abreu, “The Sensoria Reference Modelling Language,” Rigorous Software Engineering for Service-Oriented Systems, Springer, 2011, pp. 61-114.

[26] B. Tsai, W.T. Huang, Q. Chen, Y. Paul, and R. A. Xiao, “SOA collaboration modeling, analysis, and simulation in PSML-C,” IEEE International Conference on e-Business Engineering, IEEE, Oct. 2006, pp. 639-646.

[27] The official website of AMALTHEA project, [Online]. Avaliable: itea2.org/project/index/view/?project=10015, [Retrieved: Jun, 2013].

[28] M. Mernik, J. Heering, and A.M. Sloane,” When and How to Develop Domain-Specific Languages,” ACM Computing Surveys, vol. 37, Dec. 2005, pp. 316-344.

[29] S. Vinoski, “An overview of middleware,”Reliable Software Technologies-Ada-Europe, Springer, June 2004, pp. 35-51.

[30] D. C. Schmidt, “Guest Editor’s Introduction: Model-driven engineering,” Computer, vol. 39, no. 2, Feb.2006, pp. 25-31.

[31] E. Di Nitto and D. Rosenblum, “Exploiting ADLs to specify architectural styles induced by middleware infrastructures,” Proceedings of the 21st International Conference on Software engineering, ACM, May 1999, pp. 13-22.

[32] B. Kitchenham, S. Linkman, and D. Law, “DESMET: a methodology for evaluating software engineering methods and tools,” Computing & Control Engineering Journal, vol. 8, no. 3, June 1997, pp. 120 - 126.

[33] V. R. Basili, G. Caldiera, and H.D. Rombach, “The Goal/Question/Metric approach,” Encyclopedia of software engineering, John Wiley & Sons, Inc, 1994, pp. 528-532.

[34] L. Aversano, G. Canfora, A. De Lucia, and G. Pierpaolo, “Business process reengineering and workflow automation: a technology transfer experience,” Journal of Systems and Software, vol. 63, no. 1, July 2002, pp. 29-44.

[35] The official website of AUTOSAR, [Online]. Available: http://www.autosar.org/, [Retrieved: March, 2013].

[36] P. Berander and P. Jonsson, “A goal question metric based approach for efficient measurement framework definition,” Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering, ACM, Sept. 2006, pp. 316-325.

[37] W3C, “WS choreography model overview,” W3C working draft, Mar ch 2004.

[38] AUTOSAR GbR, “Software Component Template,” Ver. 4.2.0, R4.0 Rev 3, AUTOSAR Standard, 2011.

[39] AUTOSAR GbR, “Specification of RTE,” Ver. 3.2.0, R4.0 Rev 3, AUTOSAR Standard, 2011.

[40] AUTOSAR GbR, “Specification of Communication,” Ver. 2.0.1, AUTOSAR Specification, 2007.

[41] M.B. Miles and M.A. Huberman, Qualitative data analysis: a sourcebook of new methods. Sage, 1984.

Data Lifecycle Verification Method for Requirements Specifications Using a Model Checking Technique

Yoshitaka Aoki, Hirotaka Okuda, Saeko Matsuura
Graduate School of Engineering and Science, Shibaura Institute of Technology,
Saitama-City, Japan
{nb12101, ma11043, matsuura}@shibaura-it.ac.jp

Shinpei Ogata
Department of Computing, Shinshu University
Nagano-City, Japan
ogata@cs.shinshu-u.ac.jp

***Abstract*—A key to success in developing high quality software is to define valid and feasible requirements specifications to enable the production of high quality source code with minimal extra development rework. To provide invariable services to all users at any time, the data lifecycle functions of create, read, update, and delete (CRUD) are essential for handling persistent data. These important operations should, therefore, be verified at the start of development. In UML2UPPAAL, a support tool that verifies such functions, requirements specifications written in UML are transformed into finite-state automata in UPPAAL. UML2UPPAAL enables developers with knowledge of UML to benefit from the UPPAAL model checking tool without requiring UPPAAL knowledge. This paper proposes a data lifecycle verification method that uses the UPPAAL model checking tool and focuses on CRUD operations in the requirements analysis phase.**

Keywords—Verification; Model Checking; Requirements Specifications; UML; CRUD

I. INTRODUCTION

A key to success in developing high quality software is to define valid and feasible requirements specifications to enable the production of high quality source code with minimal extra development rework. Requirements specifications should have a verifiable form to guarantee their adequateness and completeness in the early stages of development. However, uncertain and ambiguous software requirements often make it difficult for developers to describe requirements specifications in verifiable form during their analysis. Although it offers insufficient verification formalization, the Unified Modeling Language (UML) [1] is a useful, common tool for formalizing requirements specifications while enabling their description in natural language. We propose a method of model-driven requirements analysis [2][3] using UML. Our method automatically generates a web user-interface prototype from a UML requirements analysis model written in activity diagrams and class diagrams. This method enables developers to confirm the validity of input and output data for each page and page transition on the system by directly operating the prototype.

Model checking has been a favored technique for improving reliability in the early stages of software development. We therefore propose a verification method in which the requirements analysis model written in UML meets essential properties that any system should meet by using the UPPAAL model checking tool [4].

Enterprise systems typically must provide invariable services to many users at a given time; therefore, the data lifecycle functions of *create, read, update,* and *delete* (CRUD) are essential for handling persistent data. These important operations should be verified at the start of development. This paper proposes a method of verifying these essential CRUD functions by using the UPPAAL model checking tool.

In UML2UPPAAL, a support tool that verifies such functions, requirements specifications written in UML are transformed into finite-state automata in UPPAAL. UML2UPPAAL enables developers with knowledge of UML to benefit from the UPPAAL model checking tool without requiring UPPAAL knowledge. This paper proposes a data lifecycle verification method that uses the UPPAAL model checking tool and focuses on CRUD operations in the requirements analysis phase.

The rest of the paper is organized as follows. Section II discusses the problems of verifying requirements specifications in terms of formalization and the applicability of model checking techniques. Section III outlines our verification method. Section IV explains UML2UPPAAL, which can be used to implement our method and support developers who have insufficient knowledge of model checking techniques. Section V describes case studies and the effectiveness of our method.

II. REQUIREMENTS SPECIFICATIONS VERIFICATION PROBLEMS

A. Problems of Writing Requirements Specifications

The primary cause of the failure of IT projects is often attributed to inadequate and incomplete requirements analysis [5]. IEEE Std 830 [6] has been recognized as a standard of requirements specifications construction. Although developers may create requirements specifications according to the standard, it is often difficult for them to fully address the interrelationship among all document components to achieve adequateness and completeness. This is because the initial requirements are written in a natural language and screen images, which are not related to the other documents in a verifiable way. Formal specification techniques, such as the Vienna Development Method (VDM) [7] and the B-method [8], provide promising approaches to formalizing requirements specifications. However, uncertain and ambiguous requirements often make it difficult for developers to describe requirements specifications in a verifiable form at the start of analysis.

UML is a promising tool for formalizing requirements specifications because of its popularity among development teams. However, step-by-step formalization is insufficient for verification. We therefore propose a verification method in which our requirements analysis model written in UML is specified as a formal description in stages by using a model checking technique.

B. Problems with Applying a Model Checking Technique

Model checking is regarded as an effective technique for improving reliability in the early stages of software development. A model checking tool uses temporal logic to model a system as a network of automata extended with integer variables, structured data types, user defined functions, and channel synchronization. Based on these properties, a system model and query expressions can be defined to specify properties to be checked. When the specified properties are not satisfied, the tool provides counterexamples that show how the properties can be falsified. The simulator helps detect the cause of defects by tracing the processes in which the counterexamples occur.

Model checking is a technique for automatically verifying a model by exhaustively checking all paths to detect properties that developers are often apt to overlook. However, because the path and state formulas should be defined by items that are used in the model, it is typically difficult for developers to define an appropriate model and formulas at all times.

Path formulas can define properties such as reachability, safety, and liveness. Reachability means that the specified state will be reached at some point in time. Safety means that something bad will never happen. Liveness means that something expected will eventually happen. State formulas need defining by expressions related to several process IDs or variables of the state.

In our requirements analysis model, a use case is defined by an activity diagram comprised of several sequences of user and system actions representing normal flows and exceptional flows in the use case. Data used in the activity diagram are classified by class diagrams for the system input/output and entity data, as shown in Figure 1. Based on a lifecycle of these entity data and actions related to them, we specify a requirements analysis model in strict descriptions to enable the automatic defining of query expressions for the model to verify the specified safety properties.

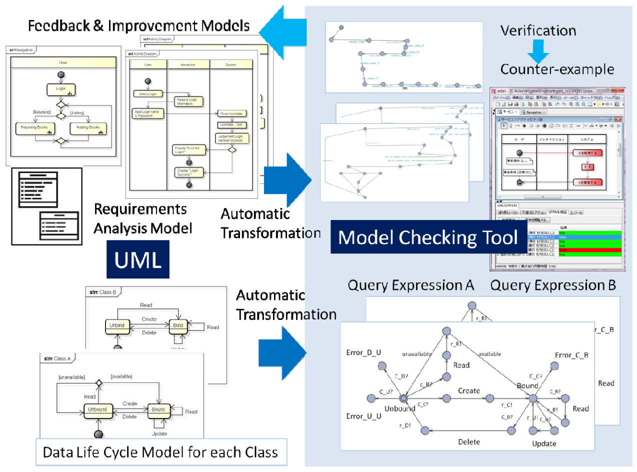


Figure 1. Verification Method using UML and a Model Checking Tool

Figure 1 shows an outline of our verification method using a model checking tool. Semi-formal UML models are automatically transformed into a network of finite automata and query expressions; these are used for producing counterexamples when the requirements analysis model has defects relating to the data lifecycle of all classes.

III. DATA LIFECYCLE VERIFICATION METHOD

A. Requirements Specifications in UML

We have proposed a method of model-driven requirements analysis using UML [2][3]. We analyze use cases and functional requirements of services. In particular, because end user needs obviously appear within the interaction between a user and system, our method proposes to clearly model the interaction.

More specifically, we identify business processes as use cases from the following questions.

- Based on the specified business rules, what types of input data and conditions are required to correctly execute the use case?
- To observe the business rule, what types of conditions should be required when the use case is not executed? Moreover, how should the system handle these exceptional cases?
- According to the above conditions, what types of behaviors are required to execute the use case?
- What types of data are outputted by these behaviors?

Based on the above questions, both business flow and business entity data, which are required for executing the target business tasks, are defined in UML by activity diagrams and a class diagram.

An activity diagram specifies not only normal and exceptional action flows but also data flows that are related to these actions. An action is defined by an action node; data is defined by an object node being classified by a class that is defined in a class diagram. Accordingly, these two kinds of diagrams enable specifications of business flows in connection with the data. This is one of the features of our method on how to use activity diagrams and class diagrams. In particular, the interaction between a user and system includes requisite various flows and data on user input, conditions, and output to correctly execute a use case.

The second feature of our method is an activity diagram that has three types of partitions: user, interaction, and system. These partitions enable ready identification of the following activities: user input, interaction between a user and system caused by the conditions for executing a use case, and the resulting output.

The third feature is a prototype consisting of web pages written in HTML that are automatically generated from the above two diagram types. The prototype, a kind of model of the final product, enables end users to clearly and easily confirm the requisite business flows in connection with the data. The generated prototype describes the required target system, except for the user interface appearance and internal business logic processing. Additionally, the prototype enables developers to confirm and understand the correspondence between their models and the final system. Developers define two kinds of diagrams based on requirements analysis from different viewpoints, such as action flows, data flows, and structure. The automatically generated prototype enables them to easily understand the consistency between their models and the target system. To facilitate a full understanding of the correspondence between each diagram and the target system, a prototype can be generated in the requirements analysis phase whenever the developer needs it. The requirements analysis model is defined using the modeling tool Astah [9].

When clients confirm that the prototype satisfactorily represents their requirements, the confirmation represents client validation that the specifications meet their expectations from an actual usage perspective.

B. Data Lifecycle Model Definition in UML

It is important that developers can verify the specifications to confirm their feasibility. To accomplish this objective, developers must confirm that a sequence of actions and data flows within the system partition of the activity diagrams can produce the expected output data from the specified input. The system-side prototype helps developers confirm the following facts.

- Input data being defined by the user can be transformed into entity data of the system.
- The existing entity data that should be generated via the other use cases and above-mentioned entity data can generate the target output data following the specified action sequence.

As a result of these considerations, developers can effectively define entity classes. During this confirmation process, it is not difficult for developers to adjust actions in the system partition in accordance with CRUD actions.

An object node has the role of a variable that stores an instance being created by the *create* action in the activity diagram. The object of the verb in the CRUD function description usually relates to the object node. The verbs shown in Table I represent CRUD functions in an activity diagram. For example, CRUD functions can be represented as “create an object,” “delete the object,” “update the object,” and “get an object.” The target object node for *create* and *read* is located at the next node of the action, as shown in Figure 2.

TABLE I. VERBS FOR CRUD ACTIONS

Action Type	Verbs
Create	create, generate
Read	read, get, search
Update	update, add, insert, change
Delete	delete

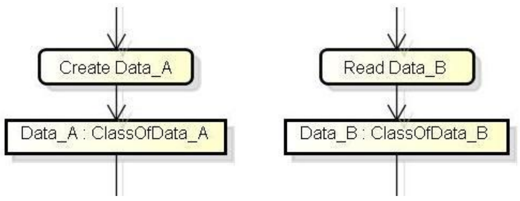


Figure 2. Relation between Object and Verb in *Create* and *Read* Actions

As a result of these adjustments, a sequence of actions in the activity diagram represents the state of changes of system entity data over the whole service by these CRUD actions.

On the other hand, entity data itself should satisfy the data lifecycle constraint of a class. For example, to *update* or *delete* an object, the object node must be bound in advance to some concrete instance object.

These essential properties of entity data are defined by using a state machine diagram in UML, as shown in Figure 3. A state machine diagram consists of several states that must be distinguished and transitions among these states. Each transition is executed by an event, if necessary, when some guard conditions are satisfied.

In this paper, we intend to distinguish whether or not each entity data is binding to an instance object, so that the system defined by the whole of activity diagrams can guarantee the correct execution of use cases in accordance with the CRUD data lifecycle.

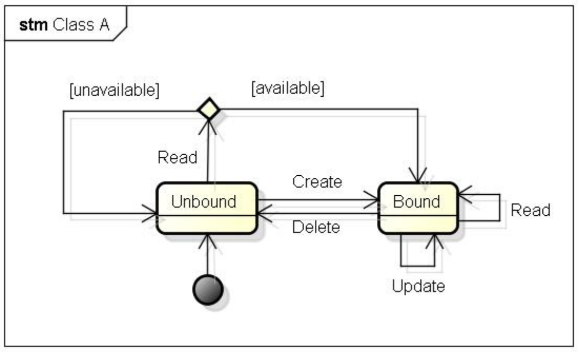


Figure 3. CRUD Data Lifecycle of a Class

Figure 3 shows a basic data lifecycle of a class. A state machine is defined for each class and named by the class. The initial state of each instance object in Class A is "unbound." After *create*, the state is changed to "bound." If the state is "bound," the instance object can accept actions such as *update*, *read,* and *delete.* If the state is "unbound" and the instance object can be obtained by the *read* action, the state is changed to "bound." If it cannot be obtained, the state remains "unbound."

However, classes do not always have the same data lifecycle. The basic state machines are therefore modified to meet the specified class. For example, if all instance objects in a class have read-only status, the data lifecycle is modified, as shown in Figure 4.

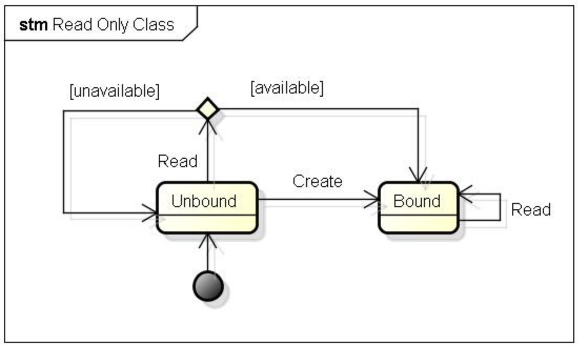


Figure 4. CRUD Data Lifecycle of a Read- Only Class

The states that should be distinguished within a class must be specified by guard conditions on the flow in the corresponding activity diagram. Figure 5 shows guard descriptions when the *read* action is executed.

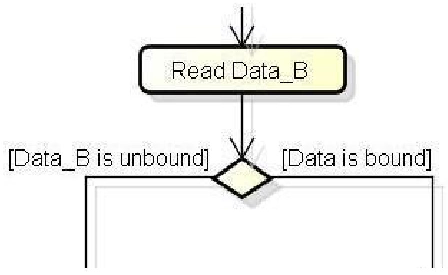


Figure 5. Guard Descriptions in an Activity Diagram

As a result, a type of CRUD action term in an activity diagram equals an event in a state machine diagram of the object in the action. A guard description equals a sentence of "<<An object>> is <<a state>>" on the control flow in the activity diagram, as shown in Tables II and III.

TABLE II. CORRESPONDENCE BETWEEN ACTION AND EVENT

Verbs of Action for an Object in Activity diagram	Event in State Machine Diagram of all Objects in a Class
create, generate	Create
read, get, search	Read
update add insert change	Update
delete	Delete

TABLE III. CORRESPONDENCE BETWEEN GUARD AND STATE

Guard description for an <<Object >>in Activity diagram	State in State Machine of all <<Objects>> in a Class
<Object> is unbound	Unbound
<<Object>>t is bound	Bound

As mentioned earlier, verifiable forms can be incrementally introduced to the requirements specifications in UML. At this point, it can be verified whether or not there are contradictions between all service flows defined in all activity diagrams and the data lifecycles of all entity objects appearing in the system partition of the activity diagram.

C. *Verification Method*

This section explains how to transform the requirements analysis model and specified data lifecycle models from UML to UPPAAL, and how to generate the query expressions.

The UPPAAL model consists of several locations and transition arrows among them, as shown in Figure 6. A location expresses a state of the system, and the transition arrow indicates several conditions named *Guard* and a sequential processing event during it named *Update*. In Figure 6, START, LOC1, and LOC2 are names of each location. "i1==0" and "i1>0" are *Guard* expressions and "flg=true" and "flg=false" represent *Update* expressions.

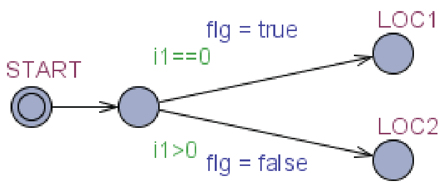


Figure 6. Basic Components of the UPPAAL Model

The requirements analysis model includes all use cases of a target system and a navigation model to integrate them. Figure 7 shows the entire structure of transforming UML models into UPPAAL models and query expressions.

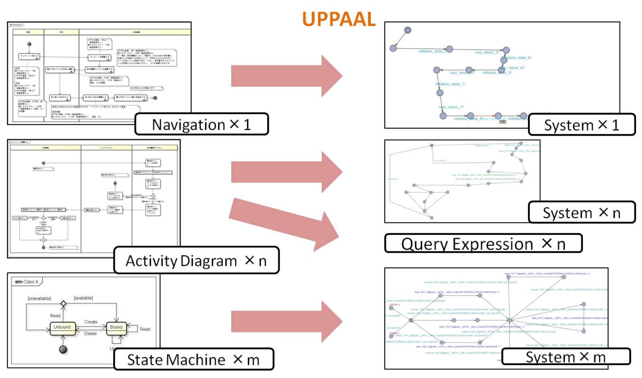


Figure 7. Transformation of UML to UPPAAL

Firstly, each activity diagram corresponding to a use case is transformed into one system model in UPPAAL. In this model, a CRUD action is transformed into a transition of three locations with channel synchronization.

Figure 8 shows the correspondence between a flow in an activity diagram and a transition in a UPPAAL system model. All nodes, such as action, object, decision, merge, start, end, and so on, are transformed into locations in UPPAAL. The control flow and data flow are each transformed into transitions, except for CRUD actions.

For example, the *create* action is transformed into a transaction sequence of three locations. The first location represents a pre-state of calling the *create* action, and the second location represents a state of creating. The third location represents a post-state of creating. The first transition flow has a synchronization channel named "c_C!" and the second transition flow has a synchronization channel named "r_C?" "c" denotes "call" and "r" denotes "return," respectively.

These synchronization channels synchronize with other channels in a system being transformed from a state machine diagram of the corresponding object class. In this case, the corresponding object means that it is an objective word of the *create* action.

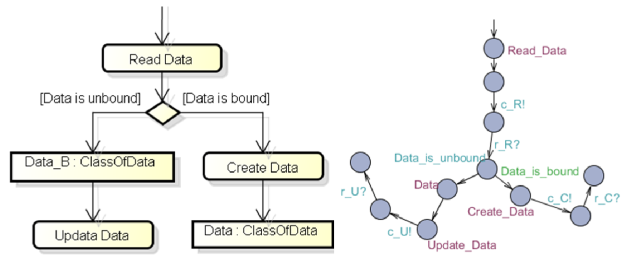


Figure 8. Activity Diagram and the Corresponding UPPAAL Model

A state machine diagram in Figure 3 is transformed into the UPPAAL model in Figure 9.

Two states are transformed into the locations named "Unbound" and "Bound," respectively. Each transition is transformed into a transition sequence of three locations, in the same way that CRUD actions are transformed. However, the channel in this model fires by calling from the system relating to the activity diagram. In this case, the first transition is fired by the corresponding object channel "c_C!" After creating, the channel "r_C!" synchronizes the channel "r_C?" in the caller system.

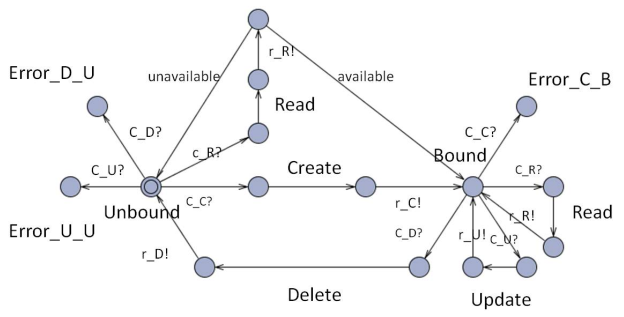


Figure 9. Transformed Data Lifecycle

A state machine diagram defines the data lifecycle of a class by using restricted actions, such as CRUD. It specifies all behaviors that all objects in the class can perform. That is, it specifies negative properties that should never happen. The state machine diagram in Figure 3 specifies that the *update* and *delete* operations should not be applied to it if an object is unbound.

The state that will never happen is then designated in the transformed UPPAAL model, as shown in Figure 9. Error_D_U, Error_U_U, and Error_C_B denote the impossible states. These states are defined for every object appearing in all activity diagrams.

As a result, we can automatically define query expressions on safety property in accordance with these models as follows.

A[] not Error_D_U_<<Object>>
A[] not Error_U_U_ <<Object>>
A[] not Error_C_B_<<Object>>

Because all names of locations in the UPPAAL model are defined by the original nodes in the activity diagrams, query expressions for the reachability property can also be automatically generated.

A navigation model integrates all activity diagrams according to the pre-conditions and post-conditions, which are a combination of several labels being added to the start or end nodes in each activity diagram. According to these conditions, all system models transformed from the activity diagrams are integrated as a UPPAAL model.

IV. UML2UPPAAL

UML2UPPAAL is a support tool that implements the above-mentioned verification method. Figure 10 shows the architecture of UML2UPPAAL, which is implemented as a plugin of the UML modeling tool Astah.

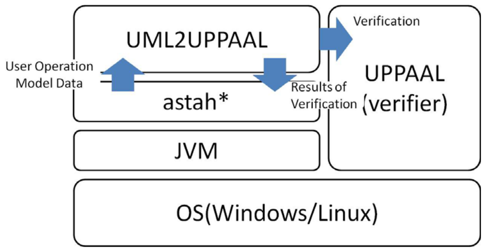
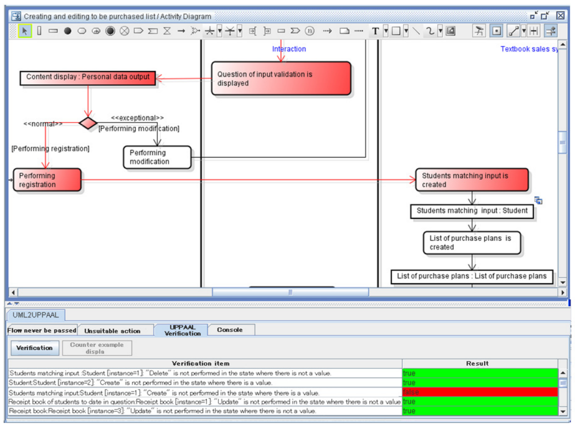


Figure 10. UML2UPPAAL Architecture


Creating and editing to be purchased list / Activity Diagram
Interaction
Textbook sales sy
Content display : Personal data output
Question of input validation is displayed
<<normal>>
<<exceptional>>
[Performing modification]
[Performing registration]
Performing modification
Performing registration
Students matching input is created
Students matching input : Student
List of purchase plans is created
List of purchase plans : List of purchase plans
UML2UPPAAL
Flow never be passed
Unsuitable action
UPPAAL Verification
Console
Verification
Counter example displa

Verification item	Result
Students matching input :Student [instance=1]: "Delete" is not performed in the state where there is not a value.	true
Student:Student [instance=2]: "Create" is not performed in the state where there is a value.	true
Students matching input:Student [instance=1]: "Create" is not performed in the state where there is a value.	false
Receipt book of students to date in question:Receipt book [instance=1]: "Update" is not performed in the state where there is not a value	true
Receipt book:Receipt book [instance=3]: "Update" is not performed in the state where there is not a value.	true

Figure 11. UML2 UPPAAL

After defining a requirements analysis model using Astah, a developer can verify it with the same tool environment. As shown in Figure 11, the result of the verification is presented by highlighting the defective items in the model. The results of executing the query expressions are shown in the lower part of the screen. During this work, developers are not required to have knowledge of UPPAAL; they only need knowledge of UML to use UML2UPPAAL and obtain the benefits of the UPPAAL model checking tool.

V. CASE STUDIES

A. *Outline of Case Studies*

We conducted a case study to evaluate the effectiveness of our method. First, five graduate students modified their UML models of the following four systems. The modifications were performed to maintain the rule of the descriptions of CRUD actions in an activity diagram. The first two systems are the currently running systems in our university. Table IV shows the scale of each model.

- Group work support system for project-based learning (PBL): GWSS
- Learning Management System: LUMINOUS
- University co-op text book sales system: COOP
- Laboratory library management system (two types): Library1, 2

TABLE IV. SCALE OF MODELS

Model	COOP	GWSS	LUMINOUS	Library1	Library2
Number of Classes	110	162	58	33	45
Number of Attributes	387	157	112	91	125
Number of Use case	7	8	8	5	6
Number of Actions	391	315	183	119	138
Average of Cyclomatic Numbers	22.9	28.2	14.9	15	12.3
Average of Number of Flows and Actions	106.5	85.7	56.1	64.3	58
Average of Number of Model elements	65.5	60.5	39.5	43.5	39.57

B. *Verification Results*

Next, the experimenters defined data lifecycle models for the specified entity data by using state machine diagrams. Having minimal knowledge of UPPAAL, they could find 83 defects in their models. The main defects found by this experiment were:

- Ten omissions of defining proper guard conditions against the nondeterministic property on the *Read* action.
- Two mistakes involving the impossible actions of *Update* and *Delete* being applied to unbounded objects.
- One mistake caused by complicated flows in which some objects could not create during the service because the position of the *Create* action was incorrect.

A navigation model is typically useful for generating a prototype system so that a user can operate it simultaneously with the final product. However, there were some cases in our experiment in which the pre-conditions and post-conditions affected the state of the object. As a result, at times there were objects of the same class but from a different data lifecycle in the activity diagram. A data lifecycle was defined for each class; however, it was necessary to adjust the state machine for the effects of the pre-conditions on the target object.

Moreover, there were instances when a complicated use case caused defects in the data lifecycle because loops occurred in an activity diagram at least two times.

It therefore must be considered that the association between classes affects the data lifecycle.

VI. RELATED WORK

Several researchers have proposed respective formal approaches to verifying specified features in the early stages of software development. Yatake [10] verified that all object states satisfy the invariant conditions between collaborative object behaviors by using a theorem-proving system. However, it requires a large quantity of strict definitions to clarify all the actions and data relating to the invariant. It is generally difficult to perform such strict work during a changeable phase, such as requirements analysis.

It is important to conduct stepwise specifications refinement by checking several verifiable features in the early stage of software development. Choi [11] proposed a verification method of the consistency between the page transition specification on a web-based system and the flow chart defining the process streams. We have also proposed a common verifiable feature in enterprise systems, such as the conditions for CRUD of entity data. Moreover, we can automatically generate the query expressions.

Achenbach [12] compared the abstraction techniques in various model checking tools and applied these tools to real-world problems. For example, the open/close behavior of the file I/O stream was modeled using the transition between states such as open, close, and error. This approach is very similar to ours. However, unlike our approach, this paper did not discuss the method on the assumption that the requirements specifications have been validated by the clients.

Several researchers have proposed support methods to effectively use model checking tools [13][14][15].

Trcka [13] proposed a method to verify the nine predefined query expressions using a Petri net, which can specify behaviors such as read, write, and delete. This study may be similar to our method. However, because query expressions depend on the properties specified by state machine diagrams, our method can be extended to verify the other properties.

Several studies [14][15] have proposed a method to transform UML models into process or protocol meta language (PROMELA) for using the model checking tool SPIN. However, because developers need to directly operate the model checking tool, they are required to have knowledge of both UML and SPIN. It is convenient that UML2UPPAAL can be used only with knowledge of UML.

VII. CONCLUSION

This paper proposed a verification method of requirements specifications in UML in the beginning phase of development using a model checking technique. UML2UPPAAL is a support tool for verifying the entity data lifecycle by transforming requirements specifications written in UML into finite automata in UPPAAL. A key attribute of UML2UPPAAL is that developers with knowledge of UML can benefit from the UPPAAL model checking tool without having UPPAAL knowledge. We are planning to apply our method to verify a security policy for requirements specifications [16] based on the Common Criteria [17] for Information Technology Security Evaluation, which is an international standard (ISO/IEC 15408) for computer security certification.

ACKNOWLEDGMENT

This work has been conducted as a part of the “Research Initiative on Advanced Software Engineering in 2012” supported by the Software Reliability Enhancement Center (SEC), Information Technology Promotion Agency (IPA), Japan.

REFERENCES

[1] UML, http://www.uml.org/,[retrieved: July, 2013].

[2] S. Ogata, and S. Matsuura, “A UML-based Requirements Analysis with Automatic Prototype System Generation,” Communication of SIWN, Vol.3, Jun. 2008, pp.166-172.

[3] S. Ogata. and S. Matsuura, “A Method of Automatic Integration Test Case Generation from UML-based Scenario,” WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS, Issue 4, Vol.7, Apr 2010, pp.598-607 .

[4] UPPAAL, http://www.uppaal.com/, [retrieved: July, 2013]..

[5] Standish Chaos Report, http://blog.standishgroup.com/

[6] IEEE Computer Society, IEEE Recommended Practice for Software Requirements Specifications, IEEE Std 830 (1998).

[7] VDMTools, http://www.vdmtools.jp/ , [retrieved: July, 2013].

[8] K. Lano and H. Haughton, “Specification in B: An Introduction Using the B Toolkit”, Imperial College Press, 1996

[9] astah*, http://www.change-vision.com/,[retrieved: July, 2013].

[10] K. Yatake, T. Aoki and T. Katayama, “Collaboration-based verification of Object-Oriented Models”, Computer Software, Vol.22, No.1, 2005, pp.58-76. (in japanese)

[11] E. Choi, T. Kawamoto, and H. Watanabe, “Model Checking of Page Flow Specification”, Computer Software, Vol.22, No.3, 2005, pp.146-153. (in japanese)

[12] M. Achenbach and K. Ostermann, ”Engineering Abstractions in Model Checking and Testing”, Source Code Analysis and Manipulation, Proc. of .SCAM ’09.,2009, pp.137-146

[13] N. Trcka, Wil M.Aalst, and N.Sidorova., “Data-Flow Anti-Patterns: Discovering Dataflow Errors in Workflows,” Proc. of the CAiSE 2009, 2009, pp.425-439.

[14] P. Bose, “Automated translation of UML models of architectures for verification and simulation using SPIN,” Proc. of the ASE, 1999, pp.102-109.

[15] L. Jing, L. Jinhua, and Z. Fangning, “Model Checking UML Activity Diagrams with SPIN,” Proc. of the CiSE 2009, 2009, pp.1-4.

[16] A. Noro and S. Matsuura, “UML based Security Function Policy Verification Method for Requirements Specification”, Proc of 2013 IEEE 37th International Conference on Computer Software and Applications, 2013, pp.832-833.

[17] Common Criteria, “CC/CEM v3.1 Release4”, http://www.commoncriteriaportal.org/cc/,[retrieved: July, 2013].

Service Relationships Management for Maintenance and Evolution of Service Networks

Aneta Kabzeva, Joachim Götze, Thomas Lottermann, and Paul Müller
Integrated Communication Systems (ICSY), University of Kaiserslautern, Germany
Email: {kabzeva, j_goetze, t_lotterm09, pmueller}@informatik.uni-kl.de

***Abstract*—The Service-Oriented Architecture (SOA) paradigm is broadly accepted for the realization of business capabilities. Hence, the maintenance and evolution of Service Networks (SN) as systems comprising multiple service-based applications is becoming a growing issue. The larger a service inventory grows and the more often services are reused, the more consequences a service change or fault can cause on related applications in the SN. While reducing the adaptation complexity of a single solution, the realization of business processes as service compositions introduces logical relations defined implicitly between the technically independent services. To preserve the consistency in the whole SN, maintenance and evolution processes have to consider all relations to the changing configuration item. We present a framework for collection, validation, and representation of service relationship information. Contributions of the proposed solution include a semi-automatic approach for relationship identification, a mechanism for completeness and consistency validation, and a tailor-made representation of relations according to stakeholder needs.**

***Index Terms*—service-orientation; service networks; service relationships; maintenance; evolution**

I. Introduction

Nowadays, Service-Oriented Architecture (SOA) is the main paradigm applied for the flexible integration of heterogeneous applications. With the introduction of the core concept of a *service*, SOA aligns the development of business processes and the underlying IT infrastructure. From a business perspective, the decomposition of business processes into reusable services allows for easy recognition of relevant software components in case of changing product requirements and better overview of IT investments for the introduction of new business capabilities. For software architects, the realization of systems as service compositions means the definition of loosely coupled units of logic accessible through a standardized interface [8]. Thus, fast adaptation to changing business requirements is achieved through the modification or replacement of the service representing the relevant business task.

While the adoption of SOA reduces the complexity of single system adaptation, the structural complexity of a Service Network (SN) as a system of service systems [5] is increasing considerably. Therefore, *maintenance* (the modification of a service-based application for fault correcting or quality improvement changes of existing configuration items) and *evolution* (the introduction of new processes, services, and policies) are growing research issues [19]. Three main factors are identified for causing the increased complexity: the increased number of configuration items that can be considered for maintenance and evolution, the existence of implicitly defined dependencies, and the restricted administrative control on some resources within the landscape.

Increased number of configuration items: the decomposition of software solutions into a number of services and the definition of the expected documents describing their interfaces, compositions, and regulations increases the set of items which need change control [23].

Implicit dependencies across applications: the reusability of services in different processes speeds up new product implementations. Yet, it leads to an increasing number of relations between services in the context of these processes. Each service reuse generates hidden chains of dependencies [10]. These dependencies are not explicitly defined [16] and affect the maintenance and evolution of SNs.

Restricted administrative control: the standardized service access through uniform interfaces allows easy integration of third-party services. Such integration introduces configuration items which are under external administrative control and are needed for the proper functioning of applications. Changes conducted by the external providers cannot be controlled and can cause disruptions of client applications [27].

To exploit the agility provided by SOA, SN operators have to deal with the resulting complexity and assure consistent landscape state after maintenance and evolution changes. The loose coupling property of services provides only technical independence [26]. The modification of a service can still affect numerous processes and applications using the service. In the context of SNs, a change can cause not only functional but also non-functional faults such as the violation of contract clauses [33]. Proper propagation of an evolutionary or maintenance change through the entire Service Network requires a rigorous knowledge of the relationships resulting from service composition and reuse. A relationship between two entities can be a functional dependency or a non-functional requirement. The relationship management solution proposed here provides a means for collecting this knowledge, validating the completeness and consistency of the collected relationship information, and presenting it in a tailor-made form suitable for the analysis needs of both business and IT stakeholders. Based on predefined patterns and constraints, it calls attention to missing relationships and inconsistencies of specifications, yet leaves the correcting actions to human interaction. To achieve this goal, several steps have to be taken [15]:

- Understand the characteristics of services residing on the different abstraction layers between business and IT, as well as the possible relationships between them to provide a basis for a completeness and correctness check of the collected information.
- Define a common format for relationship representation allowing a uniform specification of all identified relationship variations, independent from the heterogeneous specification languages applied in the realization of service-based applications.
- Design an architecture capable of supporting the collection, validation, and representation of an appropriate set of relationships relevant for SNs according to their stakeholder needs.

This paper focuses on the last step and describes an architecture, a prototype, and an exemplary case study for a relationship management framework. The remainder of the paper is organized as follows: Section 2 gives an overview of currently existing solutions for relationships management for SOA. Section 3 identifies the relationship types considered for the proposed solution. Section 4 explains the proposed framework architecture. A prototypical realization is presented in Section 5. Section 6 describes the application of the framework in an exemplary case study. Finally, Section 7 concludes the paper with a short summary and an outlook on future work.

II. Related Work

The work on relationships in service-based applications found in the literature differs according to the purpose for relationship assessment and the considered set of relationships. Regarding the purpose for relationships, existing approaches separate in two general groups: providing support for business-specific purposes [3][24] or for IT-specific purposes [1][6][27][33]. The transfer of business requirements to the executable IT services [3] and automatic process model creation [24] are the main goals pursued from the business perspective. Regarding the type of relationships, these solutions are mainly interested in the mappings between business capabilities and the IT services responsible for their execution. The maintenance and evolution scenarios from the IT perspective include failure detection and impact analysis [1][2][12], definition of service level agreements for composed services [20][33], and governance support [6]. These solutions extract information mainly on the relationships between executable services. While providing some detail on relationships properties and analysis features for the specific scenario, all these approaches capture a restricted set of relationships types. The collected relationship information is not applicable for additional analysis purposes. The solution proposed in this paper considers these approaches as a basis to identify what types of relationships should be supported by the framework and what validation features are needed.

Infrastructures for generic traceability support are offered in [30][32]. Similar to our solution, the STraS framework [30] foresees plug-ins to extract data from heterogeneous specifications of architectural artifacts. However, the actual capturing of

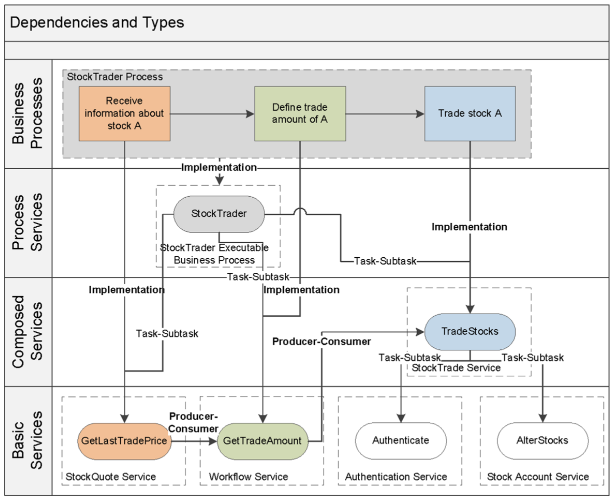


Fig. 1. Considered relationship types example

the relationships is not established. Stakeholders can query the ontology-based integrated knowledge representation according to their needs. Contrary to our approach, constraints and patterns for the validation of the collected relationships are not considered as part of the solution. The VbTrace approach [32] operates on an abstract and technology-specific process specifications. Based on a specified set of views, the View-based Modeling Framework (VbMF) produces links between view models and between elements from different view models instead of between SOA-specific architectural artifacts. From the captured relationships, the infrastructure allows code re-generation of the process implementation that should support changes in the process-driven SOA landscape. Although it considers the abstract business view on a process, this approach, and the restricted set of links that it generates, fits only a software developer's needs.

Realizing the role of knowledge management for the long-term success of service-based landscapes, several software vendor solutions for relationship management have emerged (e.g., IBM's WebSphere Service Registry and Repository [7], Software AG's SOA governance tool CentraCite [31], or Oracle's sCrawler SOA dependency tracker [28]). Typically, these solutions are built on top of a service registry and work only in combination with the corresponding vendor-specific software suite.

III. Service Relationships

The adaptability and flexibility of service-based applications is realized with the introduction of an additional *service abstraction layer* between business and application logic [8][14]. To structure the representation of both types of logic, the service layer is divided into several layers [8]. While there is a common understanding that the service layer comprises different types of services, there is no clear view what these types are. Service classification and how the different types relate to each other is normally dependent on the stakeholders' background [22]. However, independent from the number of abstraction layers defined for the realization of a SN, the aligned modification of the configuration items lying on these layers is what grants the success of a SOA and at the same time complicates its maintenance and evolution. Five maintenance and evolution challenges are identified according to the structure of SNs and addressed in the proposed solution: two address the impact estimation on business and IT change, two support the recognition of critical and wasted resources, and one regards the redundancy of services.

Business requirement change management: agile adaptation to changing requirements is the main reason for initiating a service-based application. A changing requirement has to be transferred to the execution services and will initiate changing processes on the IT side. Since business processes compose a set of business tasks, a change request for a single business task will directly trigger a change of a restricted set of services responsible for its execution. Yet, the proper functioning of services executing adjacent business tasks - which can also be part of multiple business processes composing the modified task - can also be indirectly affected.

Service change management: service quality improvement or fault correction are possible triggers for service changes coming from the IT side. Even if the service interface remains the same, a service change can have implications on the non-functional properties of the supported business capabilities. Because of the reuse of services, there can be an n to m relationship between services and tasks [30]. Multiple business tasks, and consequently business processes, can be affected.

Detection of critical services: the maintenance process is not only responsible for the execution of modifications on change requests, but also for ensuring a high quality application environment. A service is provided for consumption to an undefined set of consumers. Its usage after deployment is unpredictable for the provider. Without information on the connectivity of a service within the entire SN, it is impossible to recognize which services are crucial for the business.

Detection of wasted resources: similar to the previous challenge, service consumption is also impossible to estimate for unused services without keeping information on their connectivity. An unused service wastes storage resources or even monetary resources in the case of a third-party service. Integrating data from usage accounting can provide information about the significance of the resource waste [11].

Service redundancy prevention: a service is redundant if there is already another service offering the same functionality with the same quality for the same business capability. Services and processes are procured or provided by different stakeholders shaping the SN. A system architect cannot know all available services unless there is an explicit documentation on how the available services relate to business tasks.

To support these challenges, the proposed solution allows the extraction and explicit documentation of the relationships described below. Fig. 1 provides an example illustrating their occurrences. This figure depicts a manually created relationship model of an existing SOA-based stock trading application, which will later be considered as an exemplary case study.

Task-to-task: This relationship type represents links within the business process layer. A relationship between tasks displays the control flow (*control relationship*) or data flow (*producer-consumer relationship*) within an application. This information is explicitly available in business process descriptions and automatically extractable for relationships within a single process. A complete task-to-task relationship model for a task requires reviewing all the processes comprising the task, which is a time consuming activity without an automated relationship management solution. Task-to-task relationship information can be used to automatically map relationships between services on the executable services layers, which result in the process context and are not explicitly visible for software architects. Thus, support for service change management and the estimation of the connectivity of a service within the SN will be indirectly provided.

Task-to-service-operation: To be reusable, an executable service is usually entity-centric, defining a set of operations on a single business entity. A task within a business process defines a piece of functionality. Thus, a business task is usually executed by a single or multiple operations provided by one or more services. Providing explicit information on task-to-service operation mappings supports both business requirements change management and service change management. Because of the fine granularity of the relationship not only to a service but to its specific operation, responsible stakeholders will be able to better estimate if all related services or tasks will be affected by a change request. Furthermore, for the definition of new business processes composing an existing service task, the corresponding service can be automatically detected and prevent unwanted service redundancies. Yet, the collection of these relationships has to happen manually on the initial service selection from the software architect.

Service-operation-to-service-operation: Links within and across the executable services layers are displayed by these relationships. Service-operation-to-service-operation relationships can be captured in two ways. Relationships that are automatically collected from process services or composed service specifications residing on higher abstraction layers indicate functional dependencies (*task-subtask relationships*) across layers. Relationships that are automatically calculated from the combination on the previous two types of relationships inherit the type of the initial task-to-task relation (e.g., *producer-consumer* in Fig. 1). Both types support traceability for change management and representation of the service integration within the SN.

Business-process-to-service: Finally, if the logic modeled within a business process is controlled by an executable process service, a process-to-service implementation relation has to be explicitly captured for change management support.

IV. Service Relationship Management Framework

A. Requirements

To provide a framework for relationship management in SNs, several requirements have to be taken into account. First of all, the set of configuration items and their relationships have to be *dynamically adjustable to the target landscape*. Depending on the maturity of the SOA adoption within an organization [9], only a subset of the layers pictured in Fig. 1 may be present. A restricted set of service layers will result in a smaller set of possible relationships. The framework should not require a specific set of configuration items and relationships to be available, but adjust to the available infrastructure and its rules and constraints. Thus, an organization can adjust the completeness and correctness validations performed according to its infrastructural policies.

This leads to the second requirement: the relationship management solution should be *applicable for both existing and newly forming SNs*. To achieve this requirement, the solution should operate on available documentation without the need for modification.

Another requirement on the framework is to *capture both direct and indirect (hidden) relationships* between the configuration items of a SN. While direct dependencies can be easily extracted, either manually during design or automatically from process descriptions indirect dependencies, resulting from hierarchical service compositions and reuse in multiple processes, can be discovered only by tracing multiple descriptions, originating at different times from different stakeholders.

In highly mature SNs with repeated and augmented service compositions [9], the collection of all existing relationships can be a long running process. The more often a service participates in compositions, the higher the number of its relationships to other services will be. The more compositions in a service-based landscape exist, the higher the probability of hidden relationships. Therefore, *once calculated, relationships models should be cached and re-evaluated only on modifications*. This grants both the freshness of the model and better performance.

The relationship models acquired with the framework should be usable for different maintenance and evolution issues like change management or architecture quality analysis. Change analysis can be triggered from a modification request on a single service. A relationship model of interest, in this case, should visualize all configuration items within the infrastructure that could be influenced by the service modification. For an architecture quality analysis, the software architect can be interested in the topology of the whole infrastructure to identify business-critical services. Depending on the purpose of a stakeholder, the content of a relationship model view will differ. A *view-based representation* of the collected relationship information should be prepared from the framework to improve the usability of the models for different stakeholder groups.

B. Architecture

The architecture proposed here for relationship management in SNs comprises three horizontal layers (see Fig. 2): a *relationship collector* layer responsible for extracting relationship information from configuration item descriptions, a *relationship profiler*, which calculates additional relationships based on multiple inputs from the relationship collector, and a *relationship presenter* layer, which prepares the relationship information for stakeholder-specific extraction and visualization. A vertical layer, *relationship constraints and patterns*, supports the three horizontal layer activities through the definition of patterns and constraints specific for the structure of service-based application landscapes. The whole architecture is positioned on top of the service-based application infrastructure to be captured. It works on the basis of existing items' descriptions without requiring any specific language or additional tagging in the specifications, thus addressing the first two requirements on the desired solution.

The *collector layer* processes raw data from configuration item descriptions and transforms it into an uniform specification of the configuration item and its direct relationships according to the item-specific relationship patterns provided from the vertical layer. To support an extensible set of configuration item types, the collector layer has an extensible, modular structure. For each type of configuration item, a specialized collector module is provided that knows what type of information to search its documentation for. All recorded dependencies are presented as first-class entities in the uniform specification format and are passed for further processing to the relationships profiler layer. To be able to understand different modeling notations, like BPMN (Business Process Model and Notation) [25] or EPC (Event-driven Process Chain) [29] for business process descriptions, a set of patterns mapping the notation-specific structures to the unified information model should be provided to the collector.

The objectives of the *profiler layer* are the calculation of indirect relationships and the validation for completeness and inconsistencies. While a relationship collector processes one item description at a time, a relationship profiler combines the information from multiple descriptions. Compared to the relationship collectors, which have to be language-specific, a relationship profiler works on a unified set of data and is thus language-independent. Again, following the extensible approach advanced by the previous layer, an item-specific profiler determines how to search for relevant description files for a calculation. The completeness and consistency checks are done against item-specific constraints, defined in the vertical architectural layer on the basis of possible relationships. The completeness of the collected data is dependent on the content provided in the underlying item specifications, e.g., all services invoked in a composed service are captured as part of the landscape model. To allow for completion of mandatory information, the collection layer has to notify the responsible stakeholder and request for missing inputs, e.g., initial task-to-service operation record. In case of inconsistency, the framework only notifies the responsible stakeholder. The goal of the framework is to capture the structure of an application landscape and not its correction, which is in itself a complex issue usually requiring human interaction. To map the landscape architecture as it is, inconsistent relationships have to be kept within the model until their correction through the modification of the related configuration items. The modification will trigger a re-calculation of the

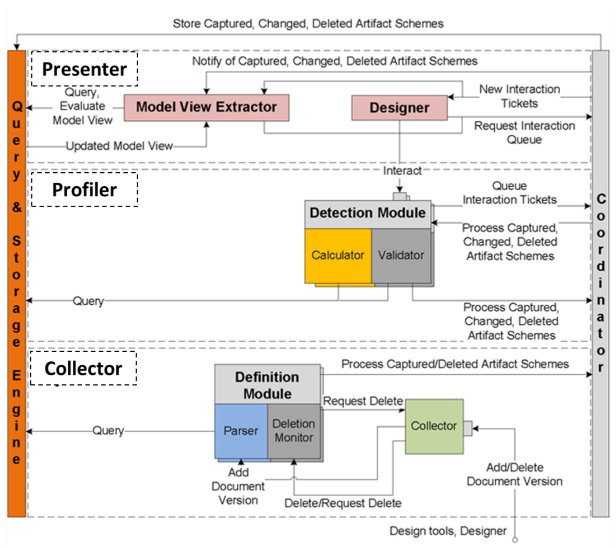

Store Captured, Changed, Deleted Artifact Schemes
Presenter
Notify of Captured, Changed, Deleted Artifact Schemes
Query, Evaluate Model View
Model View Extractor
Designer
New Interaction Tickets
Request Interaction Queue
Updated Model View
Interact
Profiler
Queue Interaction Tickets
Detection Module
Calculator
Validator
Process Captured, Changed, Deleted Artifact Schemes
Query
Process Captured, Changed, Deleted Artifact Schemes
Collector
Definition Module
Process Captured/Deleted Artifact Schemes
Request Delete
Parser
Deletion Monitor
Collector
Query
Add Document Version
Delete/Request Delete
Add/Delete Document Version
Design tools, Designer
Query & Storage Engine
Coordinator

Fig. 2. Architecture of the relationship management framework

relationships in the profiler, which will update the relationship model. The calculated relationship profiles are finally saved for stakeholder-based representation and reasoning.

The topmost *presenter layer* handles the preparation of the calculated relationship models for representation, according to the analysis-specific content requirements of a stakeholder. An automatic selection of the desired subset of relationships and configuration items needs predefined rules. Based on their role [9] in the operation of the service-based landscape, stakeholders can choose what part of a calculated relationship profile should be shown. Thereby, they should be able to restrict the visibility of configuration items as well as the type of relationships between them. The extraction of model views reduces the complexity of the model and improves its readability for stakeholders by presenting information according to their domain expertise.

V. Realization

The prototypical implementation of the framework uses the Service Component Architecture (SCA) programming model with its Apache Tuscany implementation [18] to support the development of a flexible service-based framework. The distribution of the prototype components on the three horizontal abstraction layers is depicted in Fig. 2.

The collector layer comprises a *Collector* and a set of *Definition Modules*. The collector provides an entry point for new configuration item descriptions to the framework. It acts as a central definition hub, which forwards provided definitions or configuration items deletion requests to the responsible definition modules, based on the description's type. The definition modules have the task to parse definitions of configuration items and monitor the deletion of already collected ones. The *Parser* within a definition module translates the language-specific description of a configuration item into a generic data structure which is used within the framework and marks it with a unique ID for the landscape. It also extracts notation-specific dependencies when available (e.g., task-to-task dependencies). The prototype provides definition modules for WSDL (Web Services Description Language) [4], BPEL (Business Process Execution Language) [21], BPMN, and EPC. The *Deletion Monitor* is polled every time an artifact shall be removed. When a deletion request arrives, the monitor either grants the request or throws an exception, depending on the relationship information found in the landscape model. Since only the collector is known to external design tools, it is possible to transparently integrate new definition modules for new types of configuration items for the stakeholder. No new skills or client adaptations are required in order to use the framework.

Newly captured configuration items, now represented in the generic data structure, are forwarded to the *Coordinator*. The coordinator is the central controlling unit. It forwards the configuration items to relevant detection modules for relationship profile calculation and validation. Then it sends the new information (relationships and configuration items) to the query and storage engine, and notifies the presentation components about the changes.

The functionality of the profiler layer is implemented as a set of *Detection Modules* responsible for calculating implicit relationships. Each detection module consists of a *Calculator* and a *Validator*. A calculator implements a detection algorithm for an implicit relationship type. The validation is performed in terms of completeness (whenever a component or information is missing which is needed to extract mandatory information) and consistency (whenever a potential problem embedded within the application landscape is discovered based on contradicting relationships). Each validation issue generates a ticket with a priority tag to designate the importance of its processing. The current prototype gives higher priority to consistency issues. To implement the collection of the relationships specified in section three, the prototype provides three detection modules: a mapping detection module, which collects and validates task-to-service operation and process-to-service relationships, an inter-service dependency detection module, which calculates and validates service-operation-to-service-operation relationships, and a service classification detection module capable of classifying and validating an executable service automatically. Through a concept of pluggable detection modules, the insertion of new relationship types is achieved by simply binding a new module to the coordinator.

The *Query and Storage Engine* provides an interface to the data storage containing the captured relationship models. The prototype saves the data in a DOM tree, which is managed via JDOM [13], allowing XPath processing.

The presenter layer consists of two types of clients: the *Model View Extractor*, which provides a graphical representation of the collected relationships model (cf. Fig. 3) and the *Designer*, which allows the stakeholders to contribute to the collection process. The model view extractor tool allows stakeholders to create model views based on XPath expressions and displays only a specific part of the model as a graph. The generated graphs are automatically updated whenever their

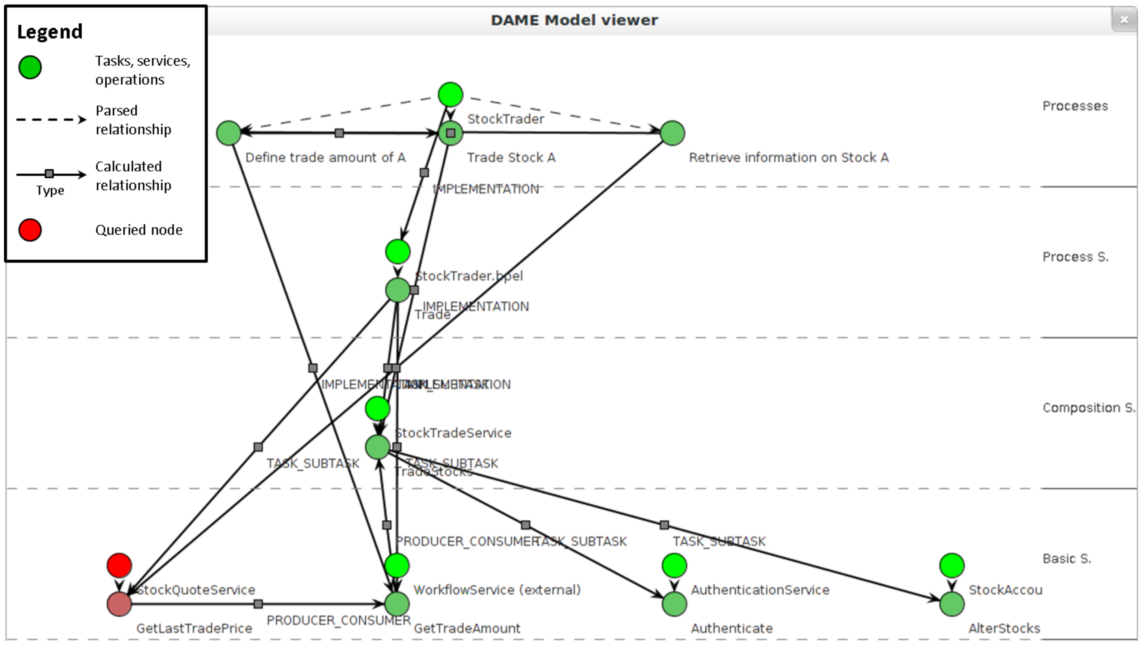


Fig. 3. Automatically generated relationships model of the stock trade application

content is affected by modifications in the architecture. The designer is a tool for manual interaction with the framework for inserting, changing, and deleting configuration items in the application landscape. In addition, it interacts with the detection modules in order to solve the validation issues recognized by the framework. Thus, the designer tool allows the framework to extract dependencies which require manual interaction as well as solve potential problems discovered within the landscape during validation.

VI. Exemplary Case Study

The proposed framework was applied to collect, validate and represent the relationships in the service network case from Fig. 1. The prototype was used to assess and validate the structure of the SN during an exemplary creation of the stock trader application. The goal was to observe the framework's behavior under conditions like missing or incomplete documentation and false service classification. For this purpose, application creation was simulated with the following steps:

1) A business analyst defines a BPMN business process description for the *StockTrader Process*.
2) A software developer defines three WSDL descriptions of the services responsible for the implementation of the process specified in step 1 - *StockQuote Service*, *Workflow Service*, and *StockTrade Service*.
3) A software architect defines the executable BPEL process specification for the *StockTrader Process*.
4) A BPEL specification of the *StockTrade Service* is imported in the network.
5) The service descriptions for the *Authentication Service* and the *StockAccount Service* composed by the *StockTrade Service* are added to the network.

The result from the first step was a business process comprising three tasks with two explicit task-to-task relationships between them. Additionally, for each of the three tasks a notification concerning the missing implementation of the tasks discovered within the process description was generated.

After the second step three basic services were added to the model with no relations. Three additional notifications of unused services were received. The stakeholder was advised to free unnecessarily used resources or provide, via the Designer tool, the mapping of which business task is implemented by which service operation. The notifications were addressed by providing the implementation relations between the three tasks and services via the Designer.

The analysis of the BPEL description from step 3 resulted in adding a process service with three task-subtask and two producer-consumer relations to the model graph.

The re-validation of the model in step 4 after inserting a BPEL description for the basic *StockTrade Service* led to a classification inconsistency. Also, two unknown service descriptions referenced in the BPEL specification were reported and asked for their insertion. The analysis of the BPEL specification resulted in the automatic relocation of the *StockTrade Service* to the composition service layer.

Addressing the requests for the service descriptions for the *Authentication Service* and the *StockAccount Service* in step 5 generated the two task-subtask relations from the *StockTrade Service* to its composite services. The resulting relations model (see Fig. 3) was automatically drawn by the framework and represents all relations from the manual assessment in Fig. 1. It shows the connectivity of the *StockQuote Service* (colored in red) within the network.

VII. Conclusion and Future Work

Understanding and explicitly modeling relationships in SNs is an essential prerequisite for controlled maintenance and evolution. This paper proposed an architecture for capturing and validating explicit and implicit service relationships. The approach considers that the different configuration items in a Service Network are specified in existing heterogeneous description languages and applies a language-independent relationship specification model to store the connectivity within the landscape. Implicitly defined dependencies resulting from service composition and reuse are captured in an explicit way, providing information on the relationship type. Applying predefined rules for relationship obligation and consistency violation, the proposed solution considers validation of the captured landscape model for completeness and consistency. For every validation issue, tickets for stakeholder interaction are generated and motivate the enhancement of the landscape infrastructure. Finally, respecting multiple stakeholder roles from the business and IT domain in a SN, and their different analysis needs on the service-oriented infrastructure at place, a view-based representation of the captured information has been considered as part of the presented framework. The application of our solution in an exemplary case study providing typical descriptions for service-based applications shows that all relationships identified as helpful for both business analysts and software architects for the decision making process during change management are captured automatically by the framework by complete landscape documentation. Incomplete documentation is discovered by the framework and reported to the relevant stakeholders.

Next steps to further improve the relationship management approach include testing of the framework capabilities and extending the validation range. Evaluations against the SAP R/3 [17] processes should assess the behavior of the prototype in a more complex service-based landscape with hundreds of processes and services. The EPC definition module necessary for this purpose is already implemented and integrated within the prototype. To increase the validation range, an exhaustive set of relationship patterns and constraints based on the architectural peculiarities of service-based infrastructures will be elaborated and integrated in the solution.

References

[1] S. Basu, F. Casati, and F. Daniel, ”Toward Web Service Dependency Discovery for SOA Management,” IEEE International Conference on Services Computing (SCC 2008), Honolulu, 2008, pp. 422-428.

[2] L. Bodenstaff, A. Wombacher, M. Reichert, and R. Wieringa, ”MaDe4IC: An Abstract Method for Managing Model Dependencies in Inter-Organizational Cooperations,” Service Oriented Computing and Applications, vol. 4, no. 3, 2010, pp. 203-228.

[3] S. Buchwald, T. Bauer, and M. Reichert, ”Bridging the Gap Between Business Process Models and Service Composition Specifications,” Service Life Cycle Tools and Technologies: Methods, Trends and Advances, 2011, pp. 124-153.

[4] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana, Web services description language (WSDL) 1.1, W3C submission, 2001.

[5] O. Danylevych, D. Karastoyanova, and F. Leymann, ”Service networks modelling: An SOA & BPM standpoint,” Journal of Universal Computer Science, 2010, pp. 1668-1693.

[6] P. Derler and R. Weinreich, ”Models and tools for SOA governance,” Lecture Notes in Computer Science, vol.4473. Springer Verlag, Berlin, Heidelberg, 2007, pp. 112-126.

[7] C. Dudley, L. Rieu, M. Smithson, T. Verma, and B. Braswell, WebSphere Service Registry and Repository Handbook, IBM Redbooks, 2007.

[8] T. Erl, Service-Oriented Architecture (SOA): Concepts, Technology, and Design, Prentice Hall, 2005.

[9] T. Erl, S.G. Bennett, C. Gee, R. Laid, A.T. Manes, R. Schneider, L. Shuster, A. Tost, and C. Venable, SOA Governance, Prentice Hall, 2011.

[10] S. Frischbier, A. Buchmann,and D. Pütz, ”FIT for SOA? Introducing the F.I.T.-Metric to Optimize the Availability of Service Oriented Architectures,” Second International Conference on Complex Systems Design and Management (CSDM 2011), Paris, France, 2011, pp. 93-104.

[11] J. Götze, T. Fleuren, B. Reuther, and P. Müller, ”Extensible and scalable usage accounting in service-oriented infrastructures based on a generic usage record format,” 6th International Workshop on Enhanced Web Service Technologies, ACM, 2011, pp. 16-24.

[12] M.A. Hirzalla, A. Zisman, and J. Cleland-Huang, ”Using Traceability to Support SOA Impact Analysis,” IEEE World Congress on Services, Washington, DC, 2011, pp. 145-152.

[13] J. Hunter and B. McLaughlin, JDOM, http://jdom.org/, 2012.

[14] N.M. Josuttis, SOA in Practice, O’Reilly, 2007.

[15] A. Kabzeva and P. Müller, ”Toward Generic Dependency Management for Evolution Support of Inter-Domain Service-Oriented Applications,” European Conference on Service-Oriented and Cloud Computing (ESOCC 2012) PhD Symposium, Bertinoro, Italy, 2012, pp.35-40.

[16] A. Keller and G. Kar, ”Determining service dependencies in distributed systems,” IEEE International Conference on Communications (ICC 2001), Helsinki, Finland, 2001, pp. 2084-2088.

[17] G. Keller and T. Teufel, SAP R/3 Process Oriented Implementation, Addison-Wesley Longman Publishing Co., Boston, MA, USA, 1998.

[18] S. Laws, M. Combellack, R. Feng, and H. Mahbod, Tuscany SCA in Action, Manning Publications Co., 2011.

[19] G.A. Lewis and D.B. Smith, ”Service-oriented architecture and its implications for software maintenance and evolution,” Frontiers of Software Maintenance (FoSM 2008), Washington, DC, 2008, pp. 1-10.

[20] A. Ludwig and B. Franczyk, ”COSMAAn Approach for Managing SLAs in Composite Services,” ICSOC 2008, Springer-Verlag Berlin Heidelberg, 2008 (LNCS 5364), pp. 626-632.

[21] OASIS, Web Services Business Process Execution Language Version 2.0. OASIS Standard, 2007.

[22] P. Offermann, C. Schröpfer, O. Holschke, and M. Schönherr, ”SOA: The IT-Architecture behind Service-Orientation,” Workshop MDD, SOA and IT-Management, Oldenburg, Germany, 2007, pp. 1-11.

[23] Office of Governance Commerce (OGC), ITIL v3: Information Technology Infrastructure Library Version 3, volume 1-5. London: The Stationary Office, 2007.

[24] A.M. Omer and A. Schill, ”A Framework for Dependency Based Automatic Service Composition,” Business Process Management Workshops (BPM 2008), Milano, Italy, 2008, pp. 535-541.

[25] OMG, Business Process Model and Notation (BPMN) Version 2.0, OMG Specification, 2011.

[26] M.P. Papazoglou, ”Service-Oriented Computing: Concepts, Characteristics and Directions,” 4th International Conference on Web Information Systems Engineering (WISE 2003), Rome, Italy, 2003, pp. 3-12.

[27] L. Pasquale, J. Laredo, H. Ludwig, K. Bhattacharya, and B. Wassermann, ”Distributed cross-domain configuration management,” Service-Oriented Computing, Springer, pp. 622-636.

[28] S. Phukan, sCrawler: SOA Dependency Tracker, Oracle Technology Network, 2009.

[29] A.W. Scheer, O. Thomas, and O. Adam, ”Process modeling using event-driven process chains,” Process-Aware Information Systems: Bridging People and Software through Process Technology, 2005, pp. 119-145.

[30] S. Seedorf, K. Nordheimer, and S. Krug, ”STraS: A Framework for Semantic Traceability in Enterprise-wide SOA Life-cycle Management,” 13th Enterprise Distributed Object Computing Conference Workshops, 2009, pp. 212-219.

[31] Software AG, CentraSite Governance Edition, User’s Guide 7.1, 2011.

[32] H. Tran, U. Zdun, and S. Dustdar, ”VbTrace: Using View-based and Model-driven Development to Support Traceability in Process-driven SOAs,” Software and System Modeling, vol. 10, no. 1, 2009, pp. 529.

[33] M. Winkler, T. Springer, E.D. Trigos, and A. Schill, ”Analysing dependencies in service compositions,” Service-Oriented Computing ICSOC/ServiceWave 2009 Workshops, Springer, pp. 123-133.

Architectural Elements of Ubiquitous Systems: A Systematic Review

Carlos Machado
Informatics Department
UFPB
João Pessoa, Brazil
carlos@ccen.ufpb.br

Eduardo Silva, Thais Batista, Jair Leite
Computer Science Department
UFRN
Natal, Brazil
eduardoafs@ppgsc.ufrn.br,
{thais,jair}@ufrnet.br

Elisa Yumi Nakagawa
Department of Computer Systems
USP
São Carlos, Brazil
elisa@icmc.usp.br

***Abstract*—Ubiquitous systems have become an important and even essential part of our daily life. For instance, smart homes are good examples where such systems can be found. However, the design and implementation of ubiquitous systems are hard tasks, as they involve several areas of computing, as software engineering, artificial intelligence, and distributed systems. This task is even harder as there is no general reference architecture that could be used to guide the development of such systems. As a consequence, each project solves the same problem in a different way, some better than others. This paper aims at exploring, organizing, and summarizing the common, essential architectural elements of those systems. We have also investigated reference architectures for this type of systems, as these architectures are important artifacts for providing such elements. For this, we conducted a systematic review that is a technique that provides an overview of a research area to assess the amount of existing evidences on a topic of interest. As main results achieved, we have found a set of eleven elements, which appears in most of the existing systems and middlewares that can be used to define a general-use software architecture. This work could certainly contribute to a more systematized development of ubiquitous systems.**

Keywords-ubiquitous computing; systematic review; software architecture

I. INTRODUCTION

Ubiquitous computing is the term initially coined by Mark Weiser [1] when referring to computer systems available everywhere at any time. These systems are often present in our lives, in form of smart TVs, smart cars, and even whole smart homes. They are capable of automating many usual tasks and support our daily live, using concepts of artificial intelligence and distributed systems.

Lyytinen and Yoo [2] proposed a difference between ubiquitous computing and pervasive computing by defining pervasive computing as models with high coupling and low mobility, while ubiquitous computing are computing models with high coupling and high mobility. However, this distinction was not widely accepted in the literature and some works do not make distinction between these two terms. It is important to highlight these differences, since some advances in ubiquitous systems could not be applied in pervasive computing, and vice versa.

An essential part of a ubiquitous project, as in any software system, is the software architecture. This architecture encompasses a set of decisions about the software organization as its structure, interfaces, behavior, and definitions of the structural elements [3]. A software architecture is essential to guide the development of a robust system, which can evolve and change through its lifetime. To help the definition of such artifact, the concept of reference architecture was proposed. A reference architecture is a special type of software architecture that provides a common understanding of a given domain, in the case of this work, the ubiquitous systems domain [4][5].

Although a number of ubiquitous systems have been proposed and impacted several sectors of the society, there is no consensus on what are the common, essential elements of a ubiquitous systems' architecture. The understanding of what are these elements is crucial for the systematic development of new systems, as well as to the maintenance and quality of existing ones.

In this context, this paper aims to identify the main elements that constitute the architecture of ubiquitous systems and whether there is any reference architecture for this domain. To achieve this goal, we conducted a systematic review that is a technique originated from the Evidence-Based Software Engineering (EBSE) [6,7], which allows to explore, organize, synthetize, and evaluate all the contributions of a research area. A systematic review allows us to identify a variety of studies that may involve theories and concepts, technological development reports, experimental research results and many others. As main results, we have observed eleven common elements, which are present in most of existing systems and middleware, and that we identified as essential elements. These elements can be used to define a general-use reference architecture, aggregating common solutions for common problems in the ubiquitous systems development.

This paper is organized as follows: Section II presents the systematic review, from its planning to the analysis of results, focusing on the architectural elements that characterize systems for ubiquitous computing. Section III contains a discussion of the collected data. Section IV presents the threats to validity of this systematic review. Finally, Section V presents final remarks and future work.

II. Systematic Review

This systematic review was conducted in the context of software architectures for ubiquitous computing, aiming at evaluating relevant studies until March 2013. To conduct this systematic review, the process was divided into three steps, as illustrated in Figure 1: Planning, Execution, and Evaluation. In the first step, we defined the search criteria and the inclusion and exclusion criteria that were used to collect related works for ubiquitous or pervasive computing. This step was also responsible for defining what we expect to extract from the found studies. The second step consisted in the execution of the systematic review, in which was performed the search for the primary studies (i.e., conference publications, periodicals, thesis, etc.), using the planning from the first step. The second step also applied the inclusion and exclusion criteria, in order to filter the results that were relevant to this review. Finally, in the third step, the results were evaluated to extract data to formulate the answer for the research questions.

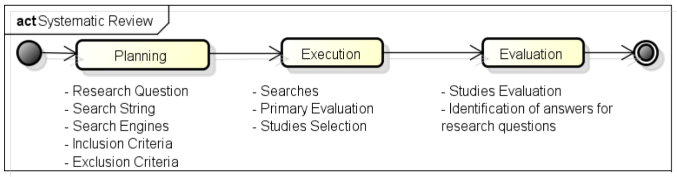


Figure 1: Systematic Review Steps

A. *Planning*

This step of the systematic review defines: (i) research questions, (ii) search strategies and (iii) inclusion and exclusion criteria.

1) *Research Questions*

In order to identify the primary studies that present common, essential architectural elements for ubiquitous systems, the following Research Questions (RQ) were defined:

- RQ1: Which are the reference architectures for ubiquitous systems? Note: This question was formulated in order to find reference architectures for ubiquitous systems. These architectures could provide common, essential elements of ubiquitous systems.
- RQ2: What are the common architectural elements for ubiquitous systems? Note: This question was defined as a complement for RQ1, and also intends to identify the common elements for ubiquitous systems.

2) *Search Strategy*

To establish the search strategy for the primary studies, from RQ1 and RQ2, the following keywords were chosen: “Reference Architecture” and “Ubiquitous Computing”. We also identified synonyms for these keywords, or similar contexts: “Reference Architecture” may be referred as “Reference Model” and it is directly related to “Software Architecture” or “Architectural Model”. In addition, “Ubiquitous Computing” is related to “Pervasive Computing”, as we explained in Section 1. Middleware for ubiquitous computing were also considered, through the keywords “ubiquitous middleware architecture” and “pervasive middleware architecture”. This inclusion had two goals: (i) to obtain an overview of existing systems, since middleware are designed to meet a wide variety of ubiquitous/pervasive applications, and (ii) the identification of the elements of these middlewares that consist in important components for ubiquitous systems. Thus, it was established the following search string: (("Reference Architecture" OR "Reference Model" OR "Software Architecture" OR "Architecture Model") AND ("Ubiquitous Computing" OR "Pervasive Computing" OR "ubiquitous middleware" OR “pervasive middleware”)). This string was used in the following publications databases: *IEEEXplorer*, *ACM Digital Library*, *Web of Knowledge* and *ScienceDirect*. The search string was adapted for each database in order to perform a directed search on title, abstract, and keywords. Only publications in English were considered.

The review process was designed as follows: The search must be performed in digital libraries, which include the main vehicles where the literature can be published. After that, the reviewers may read the title, abstract, and keywords of the found studies, in order to define which studies are worth reading the full text. After reading them, the answers of the research questions might be formulated.

3) *Inclusion and Exclusion Criteria*

To evaluate and select relevant studies, we defined a set of inclusion and exclusion criteria. These criteria were applied after each search, to define the relevance of a given study. The Inclusion Criteria (CI) was used to include relevant studies in this systematic review, namely:

- IC1: The study proposes, uses or evaluates a reference architecture for ubiquitous systems; and
- IC2: The study presents a middleware for ubiquitous computing, explicitly exhibiting its architecture.

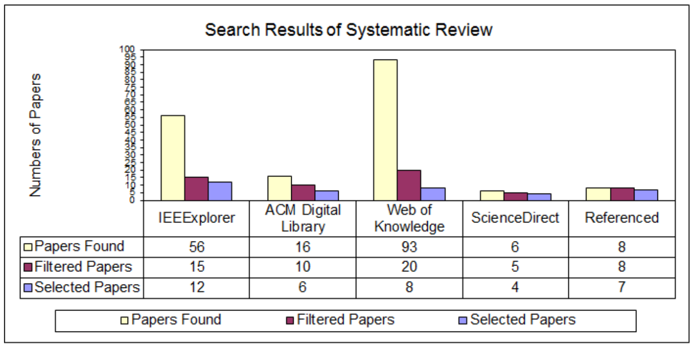
The Exclusion criteria (EC) were defined to exclude studies with no relevance for this review, i.e., studies that do not contribute to answer RQ1 or RC2. The ECs are:

- EC1: The study is not related to ubiquitous or pervasive systems;
- EC2: The study is not in English;
- EC3: The study does not have abstract or the full text is not available;
- EC4: The study consists of a compilation of studies from conferences or workshops, for example; and
- EC5: The study defines a low-level architecture, describing hardware or operational elements.

It is worth saying that a relevant study to this systematic review is defined as a study that does not satisfy any of the exclusion criteria, satisfying at least one of the inclusion criteria.

B. *Execution Results*

Upon concluding the searches, we obtained the results summarized in Figure 2. This figure shows the number of papers found by the searching process and the selected papers. In the figure, the found papers represent the number of papers returned by the automatic searching process and evaluated, i.e., we read their titles, keywords, and abstract. The selected papers represent papers whose abstracts and keywords evidenced that they are interesting for our systematic review and they were selected to be fully read.



	IEEExplorer	ACM Digital Library	Web of Knowledge	ScienceDirect	Referenced
□ Papers Found	56	16	93	6	8
■ Filtered Papers	15	10	20	5	8
■ Selected Papers	12	6	8	4	7

Figure 2: Search Results

As illustrated in Figure 2: (i) from 56 results found by the *IEEExplorer* search engine, 15 were filtered and 12 were selected for the second stage; (ii) from 16 results found by the *ACM Digital Library* engine, 10 were filtered and six were selected for the second stage; (iii) from 93 results found by the *Web of Knowledge* search engine, 20 were filtered and eight were selected for the second stage; (iv) from six results found by the *ScienceDirect* search engine, five were filtered and four were selected. Additionally, eight new studies were found from the evaluation of references of the selected articles in the first instance, and seven of them were selected. The total number of selected papers is 37. After a full analysis of each work and the application of the inclusion and exclusion criteria, 13 studies were considered relevant for our study, as listed in Table I.

Among these studies, we highlight the E6, E8, and E11 studies that present surveys on middleware for ubiquitous computing and cite, among others, precursor architectures, such as Gaia [17] and Homeros [13]. However, because these surveys have different goals we used them only as a source for searching new middlewares. Besides that, E10 presents a systematic review about ubiquitous computing, but it focuses on the characterization of ubiquitous computing projects. Note that this study is also interesting for our systematic review; however, it differs from ours, because we aim to identify the architectural elements commonly found in ubiquitous systems, as well as existing reference architectures.

TABLE I: SELECTED PAPERS LIST

Study	Author	Year
E1	*Jiehan Zhou et al* [9]	2009
E2	*Yi Liu, Freng Li* [10]	2006
E3	*Tao Xu, Bertrand David, René Chalon, Yun Zhou* [11]	2011
E4	*Shriram. R , Vijayan Sugumaran* [12]	2007
E5	*Seung Wok Han, Yeo Bong Yoon and Hee Yong Youn* [13]	2004
E6	*Saeed, A. and Waheed, T.* [14]	2010
E7	*Chang-Woo Song et al* [15]	2013
E8	*Eugster, Patrick Th.; Garbinato, Benoît; Holzer, Adrian* [16]	2009
E9	*Román, M. et al* [17]	2002
E10	*Spínola, R. and Travassos, G* [8]	2012
E11	*Raychoudhury, V., Cão, J., Kumar, M., Zhaung, D.* [18]	2013
E12	*DA, K., Dalmau, M., Roose, P.* [19]	2012
E13	*Fernandez-Montes, A., Ortega, J. A., Alvarez, J.A* [20]	2009

C. Evaluation Results

We found four studies (E1, E2, E11 and E13) that present reference architectures for ubiquitous or pervasive systems: [9], [10] [18], and [20]. The architecture proposed by Zhou [9] is focused on service composition in pervasive systems, while the architecture presented by Liu [10] was defined in a more generic way. Although the authors state that the work is about pervasive computing, the architecture of Liu [10] introduces an element of mobility, which is a typical feature of ubiquitous systems. The architecture proposed by Raychoudhury [18] was defined to support comparisons between existing pervasive systems. Thus, it does not support mobility, and it describes a multi-level structure, which blends elements of high level of abstraction, as reasoners, with elements of low abstraction, such as network protocols. Finally, the architecture proposed by Fernandez-Montes [20] is focused on building applications for smart environments, focusing on requirements for architectural elements.

These works contributed to answer RQ1 about reference architectures for ubiquitous systems. Using these four architectures and other studies on middleware for ubiquitous computing (i.e., studies E3, E4, E5, E7, E8, E9, and E12), it is possible to identify common elements that are essential for ubiquitous systems architectures, in order to find answers to RQ2. Table II describes the elements identified in the evaluated architectures.

TABLE II: COMMON ELEMENTS OF UBIQUITOUS SYSTEMS

Element	Description	Studies
Sensor	Hardware element responsible for providing context information.	E1, E3, E7, E8, E9, E11, E12
Actuator	Hardware element responsible for changing the environment, giving feedback to the user.	E3, E8
Context Service	Service used to recover context information from sensors. It may aggregate many sensors.	E1, E3, E4, E7, E8, E9, E11, E12, E13
Actuation Service	Service used to give feedback to the user. It may aggregate many actuators	E3, E8, E13
Context Repository	Data repository for context information and quality parameters	E1, E2, E3, E4, E5, E7, E9, E11, E12, E13
Event Module	Module to support asynchronous monitoring	E1, E5, E7, E9, E11, E13
Reasoning Module	Module that allow the production of new context information from existing data	E1, E2, E3, E7, E8, E9, E11, E12, E13
Adaptation Module	Module responsible for changing the system behavior according to a preset of rules.	E1, E5, E9, E11, E12, E13
Coupling and Mobility Mechanism	Mechanism that abstracts the notion of environment, making the system functional in various different environments. It uses tracking mechanisms, service search and mobile communications	E2, E4
Aggregation or Composition Module	Module for composing/aggregating context information from lower level information.	E2, E3, E7, E8, E9, E11, E12, E13
Security Module	Module responsible for implementing protection rules, such as authentication mechanisms, access restrictions and service validation.	E2, E5, E9, E11, E13

In Table II, the first column names the element, the second column contains a brief description of the element, and the third column lists the primary studies that present a concept similar or equal to the element in question. Therefore, it can be stated that for the development of ubiquitous systems, this set of eleven elements may be included, since they are commonly found in those systems. Moreover, we can conclude that they are essential elements in ubiquitous systems architectures.

III. DISCUSSION

In the context of ubiquitous systems, a related work presented a systematic review that characterized software projects for ubiquitous systems and intended to understand how this type of systems affects the life cycle of software development [8]. This study also identified a list of 10 main characteristics of ubiquitous systems, as presented in Table III. In this table, we also observe that the set of the common architectural elements found by our systematic review is able to meet the main characteristics mentioned by this previous systematic review. This table also lists the studies that present some element that aggregates a given characteristic.

It is worth highlighting that the establishment of the relationship between the characteristics and architectural elements was based on a careful analysis of this domain literature, focusing on the characteristics and roles of each element identified by our systematic review. In the next paragraph, we discuss how each characteristic is associated to the elements, as shown in Table III.

TABLE III: CHARACTERISTICS OF UBIQUITOUS PROJECTS ASSOCIATED WITH THE COMMON ARCHITECTURAL ELEMENTS OF UBIQUITOUS SYSTEMS

Characteristic	Element	Studies
Service Omnipresence	Coupling and Mobility Mechanism	E2, E4
Invisibility	Sensor	E1, E2, E7, E11, E12
	Actuator	E3, E8
	Context Service	E1, E2, E7, E11, E12
	Actuation Service	E3, E8
Context Sensitivity	Sensor	E1-E3, E7-E9, E11, E12
	Context Service	E1, E2, E7, E11, E12
	Context Repository	E1-E3, E7-E9, E11, E13
	Reasoning Module	E2, E3, E8, E9, E11-E13
	Coupling and Mobility Mechanism	E8, E9, E11, E13
Adaptable Behavior	Context Service	E1, E2, E7, E11, E12
	Event Module	E5, E7, E9, E11
	Adaptation Module	E1, E5, E9, E13
Experience Capture	Reasoning Module	E4, E11, E12
Service Discovery	Event Module	E1, E9
Function Composition	Reasoning Module	E2, E3, E8, E9, E11, E12
	Coupling and Mobility Mechanism	E8, E9, E11, E13
Spontaneous Interoperability	Coupling and Mobility Mechanism	E2, E4
Heterogeneity of Devices	Sensor	E8, E9
	Event Module	E5, E11
Fault Tolerance	Coupling and Mobility Mechanism	E4
	Event Module	E5, E9
	Adaptation Module	E1, E5, E9
	Reasoning Module	E12
	Context Service	E12
	Security Module	E11

The **Service Omnipresence** characteristic can be supported by the Coupling Mechanism and Mobility mechanism, since it uses mobile communication protocols that allow access to services anywhere, anytime.

The **Invisibility** characteristic is related to: (i) the Sensor element, which captures context information from the environment, without any explicit order of the user; (ii) the Actuator element, which forwards the system's actions to the environment; (iii) the Context Service and Actuation Service, which are the architectural elements that enable access to sensors and actuators.

Context Sensitivity is a key feature of any ubiquitous system. Sensors and Context Services are directly related to this characteristic, allowing the identification of the context and the execution of operations according to the current context. The Context Repository is responsible for storing context information. The Reasoning Module performs inferences about contextual information and can produce new information. The Aggregation or Composition module performs the context information composition.

Adaptable Behavior defines that the system must adapt to the environment, offering services according to the current context. The Context Service is essential for the identification of the context, while the Event Module is responsible for triggering an event for context changing. After that, the adaptation can be performed. This characteristic may also be attributed to the Context Repository, as in E5. Finally, the Adaptation Mechanism performs the required adaptation for the new context. The

Experience Capture characteristic consists of capturing and storing information for future use. It is typically related to the Reasoning Module, which uses machine learning and other artificial intelligence techniques. This module has a role similar to the Aggregation or Composition Module found in some studies, such as E8. The existing difference between these modules lies in the fact that the Reasoning Module is able of generating new context information, while the Aggregation or Composition Module only groups or composes the context information. In most studies; however, these modules are integrated.

Service Discovery is supported, in most studies, by the Event Module, which is proactive in relation of services, monitoring and discovering available services, making them available through a publish-subscribe mechanism. However, this behavior may be aggregated to the Context Repository, as in E5.

Service Composition determines the system ability of providing new services to the final user, based on existing services. The Reasoning Module is related to this characteristic, since this module must be able of identifying the basic services (E2, E3, E8, E9, and E12) and compose them according to some business rule. The Aggregation or Composition Module, in some studies (E8, E9, E11), is used to perform the composition. In addition, the Reasoning Module can infer new contextual information to provide it as a new service. However, the new services that may be offered vary between applications.

Spontaneous Interoperability is the system ability of using many elements without the need of external intervention. This characteristic is supported by the Coupling and Mobility Mechanism, since this element is responsible for mobile computing protocols and for handling, in a high abstraction level, environment changes (E2 and E4).

The **Heterogeneity of Devices** characteristic defines that the distinct elements must be uniformly accessed. The E8 and E9 studies discuss the role of sensors in providing information from heterogeneous sources, as well as the role of the Event Module to monitor different services in a transparent way to users.

Regarding the **Fault Tolerance** characteristic, the Coupling and Mobility Mechanism is directly related to the mobile devices used by the users to access the system. Therefore, this mechanism must be able of handling the most common problems related to mobile computing, as connection instability and fluctuations in the data flow (as shown in E4). The Event Module may trigger many events, including faults or errors in any of the available services. The faults can the handled by the Adaptation Mechanism. In E12, the responsibility of fault tolerance is diffuse, whereas several elements detect and treat its own inappropriate behavior. The Security Module also supports this characteristic, by providing authentication and access control mechanisms.

In short, it is observed that the common architectural elements identified by this study adequately meet all the characteristics of ubiquitous systems stated by Spinola and Travassos [8].

Note that although only two studies (studies E2 and E4) explicitly presented the Coupling and Mobility Mechanism, it was identified that this element type is essential for ubiquitous systems, since these systems have essentially a mobility element, to allow the system be accessible anywhere. The E3 and E7 studies presented a query mechanism to recover context information from the Context Repository. However, we chose not to explicitly insert this element, since it was observed that this element is commonly implemented as part of the Context Repository, because it is highly dependent on the format of the stored context information. Many low-level or very specialized elements were not considered common architectural elements. For example, the Operating System and Network Protocol were not considered, since they were cited only by studies about low level architecture.

IV. THREATS TO VALIDITY

A major threat to validity of this systematic review refers to the completeness of this study, i.e., if in fact all the related papers were included. This problem may have occurred because relevant studies were not found by the search mechanisms, for instance, by the technical limitations of the search mechanisms. Another threat refers to the results and conclusions presented in the evaluation step. We tried to minimize those problems by adopting a dual revision approach for each paper, performed by the different reviewers of this work. This strategy contributes to reduce possible bias or misinterpretation. The findings were also validated by more than one reviewer. These strategies ensured that the set of the found architectural elements cover the essential requirements of an architecture for ubiquitous systems.

V. Conclusion and Future Work

The ubiquitous computing enables the use of contextual information from any environment at any time. Ubiquitous computing exploits technological advances in pervasive computing and mobile computing, integrating mobility, engagement, and distribution. Considering its relevance, attention to the development of ubiquitous systems is essential.

This work presented a literature review with the aim of summarizing the knowledge about reference architectures and common architectural elements for ubiquitous systems. As main result, the common, essential elements of ubiquitous systems were identified, analyzed, and summarized. This paper also mapped these elements in the main characteristics of ubiquitous systems. This mapping is important to verify that the identified elements meet the essential characteristics of ubiquitous systems. Furthermore, this set of elements can be considered as basis of any ubiquitous systems. Therefore, the identification of this set can be considered an important contribution to systematize the development of such systems. Moreover, we have observed that the four reference architectures found in our systematic review do not comprise all architectural elements identified in this work. In this scenario, as a future work, we intend to define a more complete, well-structured reference architecture. Thus, it is intended that this architecture can effectively contribute to the development of ubiquitous systems that have become increasingly important to our daily lives.

VI. ACKNOWLEDGEMENTS

This work is supported by Brazilian funding agencies FAPESP, Capes, and CNPq.

VII. References

[1] M. Weiser. “The Computer for Twenty-Frist Century”. Scientific American, September 1991.

[2] K. Lyytinen and Y. Yoo, “Issues and Challenges in Ubiquitous Computing”. Communications of the ACM. n. 12, v. 45, 2002. pp. 63-65.

[3] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice. Addison-Wesley, Boston, 1998.

[4] R. Cloutier et al, “The Concept of Reference Architectures”. Systems Engineering, 13. V. 13, n.1, UK, 2010, pp. 14-27.

[5] E. Y. Nakagawa, P. O. Antonino, and M. Becker, “Reference Architecture and Product Line Architecture: A Subtle but Critical Difference”. Proc. 5th European Conference on Software Architecture (ECSA'2011). Essen, Germany, 2011. pp. 207-211.

[6] T. Dyba, B. Kitxenham, and M. Jorgensem, “Evidence-Based software engineering for practitioners”. IEEE Software, v. 22, n. 1, 2005. pp. 58-65.

[7] B. Kitchenham, T. Dyba, and M. Jorgensen, “Evidence-Based Software Engineering”. Proc. 26th International Conference on Software Engineering (ICSE). IEEE Computer Society. Washington, DC, USA, 2004. pp. 273-28.

[8] R. Spínola and G. Travassos, “Towards a framework to characterize ubiquitous software projects”. Information and Software Technology. v. 54, 2012. pp. 759-785.

[9] J. Zhou et al., “PSC-RM: Reference Model for Pervasive Service Composition”. Proc. Fourth International Conference on Frontier of Computer Science and Technology. 2009. pp. 705-709

[10] Y. Liu and F. Li, “PCA: A Reference Architecture for Pervasive Computing”. Proc. 1st International Symposium on Pervasive Computing and Applications, 2006. pp. 99-103.

[11] T. Xu, B. David, R. Chalon, and Y. Zhou, “A context-aware middleware for ambient intelligence”. Proc. Workshop on Posters and Demos Track. ACM, NY, USA. 2011. pp. 10:1-10:2

[12] R. Shriram and V. Sugumaran, “Adaptive middleware architecture for information sharing on mobile phones”. Proc. 2007 ACM Symposium on Applied computing. ACM, New York, NY, USA. 2007. pp. 800-804

[13] S. W. Han, Y. B. Yoon, H. Y. Youn, and W. Cho, “A new middleware architecture for ubiquitous computing environment”. Proc. 2nd IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems. 2004. pp. 117-121

[14] A. Saeed and T. Waheed, “An extensive survey of context-aware middleware architectures”. Proc. IEEE International Conference on Electro/Information Technology (EIT), 2010. pp. 1-6

[15] C. Song, D. A. Lee, K. Chung, K. Rim, and J. Lee, “Interactive middleware architecture for lifelog based context awareness”. Multimedia Tools and Applications. Springer, US, 2013. pp. 1-14

[16] P. Eugster, B. Garbinato, and A. Holzer, “Middleware Support for Context-Aware Applications”. Proc. Middleware for Network Eccentric and Mobile Applications, Springer, 2009. pp. 305-322.

[17] M. Román, C.K. Hess, R. Cerqueira, A. Ranganathan, R.H. Campbell, and K. Nahrstedt, “Gaia: A Middleware Infrastructure to Enable Active Spaces”. Proc. IEEE Pervasive Computing, 2002, v. 1, pp. 74-83

[18] V. Raychoudhury, J. Cão, M. Kumar, and D. Zhaung, “Middleware for pervasive computing: A survey”. Pervasive and Mobile Computing, April 2013, pp. 177–200,

[19] K. Da, M. Dalmau and P. Roose, “WaterCOM: An Architecture Model of Context-Oriented Middleware”, Proc. Workshops (WAINA), 2012 26th International Conference on Advanced Information Networking and Applications, 26-29 March 2012. pp. 53-60.

[20] A. Fernandez-Montes, J. A. Ortega, J. A. Alvarez, and L. Gonzalez-Abril, “Smart Environment Software Reference Architecture”. Proc. NCM '09. Fifth International Joint Conference on INC, IMS and IDC, 25-27 Aug. 2009, pp. 397-403

Architectural Decisions in the Development of Multi-Layer Applications

Jose Garcia-Alonso
Quercus Software Engineering Group
Centro Universitario de Merida
Merida, Spain
Email: jgaralo@unex.es

Javier Berrocal Olmeda
Juan Manuel Murillo
Quercus Software Engineering Group
Escuela Politecnica
Caceres, Spain
Email: {jberolm,juanmamu}@unex.es

***Abstract*—Multi-layer architectures have become one of the most widely used architectures for enterprise application development. Among other reasons, this is due to the proliferation of development frameworks simplifying the implementation of applications based on such architectures. However, the software architect is faced with a significant challenge at the beginning of the development process with having to decide among the great number of design patterns and development frameworks that support these architectures. The present work proposes a technique to assist the architect in deciding which technologies are best suited to satisfying both the functional and the non-functional requirements of the system. This technique forms part of a broader procedure to facilitate the software architect's task of converting a preliminar concept of an application into a specific design optimized to the project in hand.**

Keywords*—*Multi-layer architectures; design patterns; development frameworks; architectural knowledge.

I. Introduction

A significant proportion of applications being developed today are targeted at enterprises. They tend to be complex systems with significant scalability and performance requirements. These requirements are further complicated by the rise in recent years of cloud computing and development for mobile platforms. When these applications make use of such environments the non-functional requirements regarding reliability, performance, integration, security, migratability, etc; gain even greater relevance [1].

The focus of the present study is on the development of the back end of these applications – specifically, of those whose development is based on the use of multi-layer architectures. Defining and designing the architecture of a system of this type is an arduous and complex process for the architect.

Firstly, many frameworks and design patterns have been proposed to simplify the implementation of these architectures [2]. Currently, the use of development frameworks, and consequently of the design patterns that they help to implement, is a widely extended practice. Proof of this is the large number of available frameworks [3], the number of versions released annually, and the job offers that require their skills [4]. The great number of existing design patterns and development frameworks forces architects to devote substantial effort to learning them. It is not enough to obtain an in-depth knowledge of a set of them, it is necessary to have adequate knowledge about all of them, the web of interactions between them [5] and the use of one or another favouring or penalizing the fulfilment of certain non-functional requirements.

And secondly, in order to make these decisions properly, the architect must have a thorough knowledge of the requirements and of the relations between them. The architect must extract the knowledge about the system requirements from the analysis of a series of documents on which, in many cases, the relationship between functional and non-functional requirements are not explicitly detailed [6]. The ability of the architecture to meet the system's requirements depends on the interpretation of these documents. Therefore, any misinterpretation on her part in this complex analysis implies the inclusion of errors in the architecture.

The combination of these two factors exposes the architect to situations in which a misinterpretation could lead to the choice of an inappropriate design pattern or development framework, with the problems that it would entail [7]. The present work focuses on the architect's decision making. Its principal contribution is a technique which makes use of a feature model to provide the architect with a catalogue of the commonest architectural decisions in the development of framework-based multi-layer applications [8]. The architect can use that catalogue, alongside the preliminary design of the application marked with quality attributes [9], as a basis for orderly decision-making. The decisions actually made by the architect are also recorded and later they can be used as design guidelines in developing similar applications [10].

The rest of this communication is organized as follows. Section 2 presents the motivations for this work. Section 3 gives a complete overview of the proposal. Section 4 details the proposed decision-making process and the automatisms provided. Section 5 specifies the tools which support this proposal. Section 6 gives a review of the most significant related work. Finally, Section 7 presents the conclusions to be drawn from the study, and some indications of future work planned in this line of research.

II. Motivation

During the development of industrial software applications, the preliminary designs obtained from the requirements are not usually implemented as such. First, they must be adapted to the chosen multi-layer architecture [11].

Once the layer architectural pattern [12] has been applied to the initial design of a system, different design patterns may be used in each layer. For example, the Data Access Object (DAO) pattern can be used in the design of a persistence layer and the Model-View-Controller (MVC) pattern to design a presentation layer. This kind of multi-layer architecture has become widely

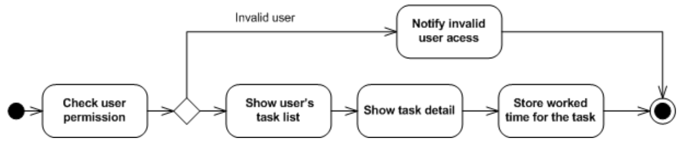


Fig. 1. Activity diagram (part of the initial design).

accepted in the industry, especially since the introduction of development frameworks [2].

However, the use of such architectures has its downsides. Specifically, what was once a clear advantage, nowadays, with the explosion in the number of frameworks and patterns, has become an additional risk. The architect needs a depth knowledge about a large number of frameworks and the interrelations between them. For this reason, the architect's work becomes more error prone, and, worse, these are errors that may have a significant impact on the overall project.

In order to motivate the problems addressed by this work we present here an example of the design process for an application's architecture. Figure 1 shows an activity diagram of a very common use case in enterprise applications. This use case allows the system's users to check a series of elements, to see detailed information about any one of them, and to modify that information. The system performs a check on whether or not the user has permission to perform that operation. If not, a notification is sent informing of an invalid access attempt.

Establishing the system requirements is the starting point for architects designing a new architecture. The designed architecture should maximize the chances of complying with all the requirements. This is in itself a complicated task. In many cases systems are asked to meet requirements that are difficult to combine and the architect should reach a balance [6].

Once the architect acquires all the information about the requirements, he or she should start its design. For this, the architect must take into account the large number of patterns available. Choosing a particular pattern can lead to different degrees of requirements being satisfied, especially in the case of non-functional requirements [13].

Referring the case study, the developed system must meet certain security and auditing requirements. If the architect omits, forgets or misinterprets the relation between this requirements, she might try to meet both requirements at the same time. However, these requirements may conflict making the architect choice a possible cause of future errors.

This study presents a technique to simplify and record architectural decisions in the development of multi-layer applications. Studies such as those of Zimmermann [10], [14] focus on the architectural decisions making process in a similar way as discussed in this article. However, to the best of the authors' knowledge, despite the industrial acceptance of multi-layer architectures and development frameworks, there has been no previous work on support for architectural decision making in framework based multi-layer applications.

This work forms part of a broader proposal that covers the entire process of designing these applications. In the next section, we shall briefly describe the complete proposal so as to provide a clear context for the contribution to be described in the rest of the paper.

III. Multi-layer Enterprise Applications

Figure 2 shows a complete diagram of the process proposed for the development of framework-based multi-layer applications.

It shows how the proposed process begins with the preliminar design, normally consisting of a use case diagram and multiple activity diagrams representing the behaviour of those use cases. In activity 1 this design has to be refined by the architect or requirements experts to include information about the quality attributes of the system.

As mentioned above, usually the relationship between functional and non-functional requirements are not explicitly detailed [6]. To make these relationships explicit, the architect or the requirements expert mark the preliminary design with information about the quality attributes to be met by the application. The technique used to accomplish this marking is described in more detail in another paper by the authors [9].

Once the architect has the marked design, the next task is to select the layers into which to split the application, activity 2 in the diagram. In order to simplify this task, the process offers to the architect an initial selection of layers. This initial selection is based on the preliminary design and the information added by the marks. However, is the architect who must refine, validate or reject it based on other criteria such as technological limitations, type of project, client, etc. This task is done in the activity 3 in the diagram.

Once the layers have been selected, the initial design can be refined to adapt it to them. This adaptation is performed by a transformation of the model that takes as input the initial design and the configuration of the feature model. This correspond to activity 4.

Feature modeling is one of the most extensively accepted techniques for modeling variability [15]. The specific model used in the present work follows the approach of Cardinality Based Feature Modeling, a widely used technique with proven usefulness in working with development frameworks [16].

To use a feature model as input or output for models transformations it needs to conform to a clearly defined structure or some sort of "metamodel". This structure must, however, be flexible enough to incorporate both the existing architectural and technological elements and any new ones that may arise in the future. For the model to have these features, we performed a study of some of the most used development frameworks (including Spring, Hiberate, Struts, JSF, CXF, Axis, DWR, etc.). More details on the analysis performed for the creation of the feature model and the decisions made for its creation may be found in [8].

At this point in the process, the architect must specify the design patterns and development frameworks on which to base the final design of the application, activity 5 in the diagram. To make this selection, the architect uses the information contained in the feature model, and then must link each element of the layer design to the architectural decisions that affect it, activity 6 in the diagram.

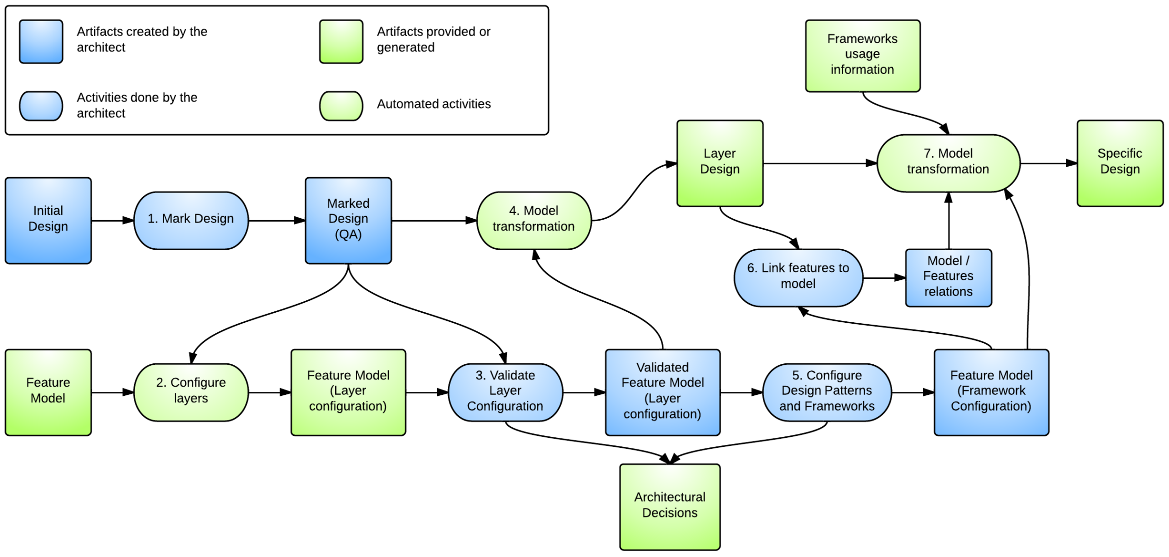

Artifacts created by the architect
Artifacts provided or generated
Activities done by the architect
Automated activities
Initial Design
1. Mark Design
Marked Design (QA)
4. Model transformation
Layer Design
Frameworks usage information
7. Model transformation
Specific Design
6. Link features to model
Model / Features relations
Feature Model
2. Configure layers
Feature Model (Layer configuration)
3. Validate Layer Configuration
Validated Feature Model (Layer configuration)
5. Configure Design Patterns and Frameworks
Feature Model (Framework Configuration)
Architectural Decisions

Fig. 2. The multi-layer application development process.

It should be noted that we propose a specific order for the decision making process, first the layers then the design patterns and finally the development frameworks. However, this order is not fixed and the architect can change it to suit their needs and preferences The abilities exhibited by features model to allow such flexibility were one of the main motivation to choose them as our architectural knowledge representation tool.

Finally, with all the information available, a model transformation is performed to convert the application layer design obtained previously into a specific design for the architectural decisions taken by the architect, activity 7 in the diagram. For this transformation, information is required about the development frameworks to be used. This information is included in the transformation by means of specific models describing the use of a particular technology.

The present work focuses on the architect's decision making. Specifically, in activities 3 and 5 in the diagram shown in Figure 2. To accomplish these activities, the architect uses three elements: the feature model containing information about the design patterns and the development frameworks that can be use for the development, the preliminary marked design that contains information about the relationship between the requirements and the system's quality attributes and his or her own knowledge about the system.

For a better understanding of this technique, we shall describe an example of how it works. Figure 3 shows the same activity diagram used previously enhanced with additional information about the quality attributes that the system must satisfy. Specifically, the verification of user permissions must meet security requirements, the notification of invalid access attempts should communicate with an external system and the modifications made by users should be auditable.

IV. Making Architectural Decisions

The elements presented in the previous section compose the basis for the architect's decision making. The present section will describe in detail the activities 3 and 5 in the diagram.

A. *Selecting layers*

A reasonable way to begin the decision making process when designing a multi-layer architecture is to choose the layers that will form part of the application. Many applications of this type use a common set of layers with similar functionality. Examples are the persistence, the presentation, and the Web service layers. The feature model we use contains a set of common layers, which can be easily expanded by adding new layers.

To simplify the architect's work, the information about the quality attributes added to the application's preliminary design can be used to offer an initial suggestion of an appropriate set of layers that might satisfy those attributes.

The layer suggestion process is based on a relatively simple set of rules. Specifically, a layer is suggested based on two criteria.

The first is the presence of certain elements in the preliminary design specific to each layer. The presence of these elements, which can be detected by querying the preliminary design model, determines whether a layer is to be proposed to the architect as part of the application's architecture. For example, the web services layer is suggested when the preliminary design includes interactions with external systems.

The second criterion is based on the marks with quality attribute information. Certain quality attributes entail the suggestion of certain layers. The presence of these marks is also detected by querying the design model. For example, whenever there appears an activity marked as Auditable the use of a log layer is suggested.

Table I shows a summary of the main criteria used to suggest the most common layers.

Technically these criteria consist of a set of model transformations that take as input the preliminary design and the

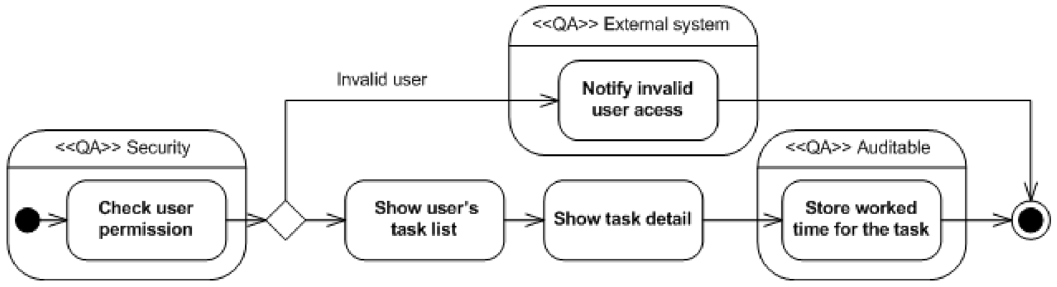


Fig. 3. Marked activity diagram.

TABLE I
LAYER SELECTION CRITERIA.

Layer	Criteria
Persistence	There is direct interaction with a Database or the same object is used in the activity diagram of more than one use case
Business logic	Always present, included here for further configuration at a lower abstraction level
Presentation	There is interaction with a human actor
Web services	There is interaction with external systems
Security	There is a Security mark on one or more of the elements in the UML diagrams
Log	There is a Maintainability mark on one or more of the elements in the UML diagrams
Test	There is a Testability mark on one or more of the elements in the UML diagrams

feature model. The output of this transformation is another model with an initial configuration of the feature model in which the suggested layers are selected.

Applying these criteria to the diagram shown in Figure 3, the architect is offered a basic initial selection of layers. This selection is presented in the form of a partial configuration of the feature model. In the case of the diagram in the figure, the architect will be proposed the use of the following layers: persistence because the activity diagram requires information to be retrieved that was stored in the system earlier and information to be stored for later use. Presentation because this layer includes all the elements related to interaction with the user. Web services because notifying an unauthorized access attempt requires communication with an external system. Security because checking the user's privileges has to be a secure task. And log because some of the diagram's activities have to be auditable.

An additional layer is suggested that encapsulates the application's business logic. This is a standard layer in enterprise applications to incorporate elements relating to the application's behaviour.

The architect's next task is to validate the set of suggested layers, or to modify it as may be deemed opportune. The final set of layers selected by the architect is registered as a partial configuration of the feature model and it is used to perform an initial model transformation. This transformation gives as output a specific design for the layers in which to split the application where each activity is represented in the layers in which it operates. Figure 4 shows a small fragment of the output of this transformation applied to a the preliminary design shown previously.

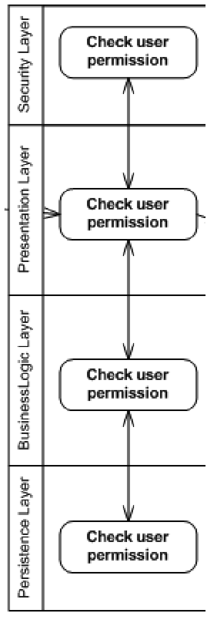


Fig. 4. Fragment of the layer adapted design.

B. Selecting patterns, technologies, and use

The following architectural decisions that have to be made consist in selecting the design patterns and technologies to use in the development of each of the layers identified in the previous section. It is possible, as was done during the selection of the layers, here too to present the architect with an initial selection based on the information contained in the initial design. Now, however, the architect's decisions have greater importance. In many situations, the choice of a particular technology will depend less on the application's requirements and more on either the criteria of the firm responsible for the development or the preferences of the architect. For example, the experience of the developers is one of the most important factors when selecting a technology to implement the MVC design pattern in a presentation layer. There are a number of frameworks that give full support to this pattern, the one which is normally used is that with which the architect or the development team has most experience. However, one should not forget that the chosen technologies must support the quality attributes of the application. So, the information about the quality attributes included in the preliminary design is most useful to validate the architect's decisions than to provide initial suggestions.

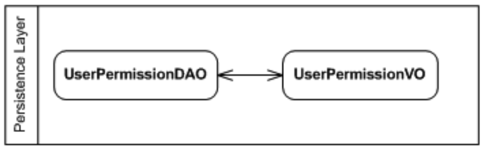


Fig. 5. Fragment of the design pattern adapted design.

Due to this, the weight of this task falls largely on the architect. It is generally done in two steps. In the first, the architect selects the design patterns to use in the development of each layer. To make this choice the architect uses the list of patterns available in the feature model for each layer and the preliminary design with the information about the functional and non-functional requirements of the application. Typically, the selection is that which can best fulfill the application's functional and non-functional requirements. However, the architect has the final say on the matter and can take architectural decision based on different criteria such us his or her own previous experience or the development team knowledge about specific technologies. In the example we are using, the architect could choose the MVC and Web Remoting patterns jointly for the presentation layer, and the ReST pattern instead of SOAP for the Web services layer. With this information, it is possible to apply a new partial transformation to obtain a more detailed design adapted to the design patterns chosen by the architect. Figure 5 shows a small fragment of the result of this transformation after applying the DAO pattern to an activity in the persistence layer.

In the second step, the architect must select which technology or development framework will be used to support each of the selected design patterns. Again, this set of architectural decisions is based on the information contained in the feature model and the preliminary design. The selection will be made from among the technologies specific to the design patterns chosen and will depend mainly on the architect experience. For example, in the case of ReST, the architect must choose a technology that will support it, omitting consideration of other Web service technologies. Also, the presence in the feature model of the constraints mentioned above prevents the architect selecting incompatible technologies, and provides suggestions of technologies that are closely related to those already chosen. With this information, it is possible to apply the last transformation to obtain design adapted to the architectural decisions. Figure 6 shows a small fragment of the result of this transformation after using the Hibernate framework to implement the DAO pattern.

As mentioned above, for clarity reason, in this paper we show how our proposal supports architectural decisions in a specific hierarchical order. However, the architect could choose a different order. The use of feature models to specify architectural knowledge and the architectural decisions make it possible whilst the consistence is kept during the transformation process.

Using the described process provides the architect with two major advantages. One is that the number of options to consider when making decisions is pruned, with irrelevant elements eliminated from the process, and allowing the architect to focus on the use of just an allowed set. The other advantage

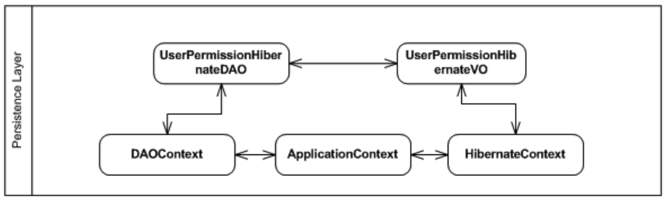


Fig. 6. Model transformation sequence.

is that using the feature model provides a simple mechanism for storing architectural decisions. Every decision made by the architect is reflected as a configuration of the feature model, and these configurations are easily stored for reference and use in future developments. The firm's architects will thus have a set of design guidelines based on successes or failures in previous projects.

V. Support Tools

In order to validate the techniques proposed in this paper, a set of tools is under development targeted at providing support to the entire process described in Section 2.

For tasks related to the architectural decision making, the core of the present work, a feature model is used that is similar to that described in Section 3, and which contains information on more than a dozen of the commonest development technologies.

Regarding the architectural decisions themselves, the techniques described in this paper are supported by a custom-designed Eclipse plug-in for the creation of multi-layer architecture Java projects. To create one of these projects, the plug-in needs a feature model such as that mentioned above. The options that will be presented to the architect for decision making are obtained from this model. The plug-in configuration allows specification of the URL at which to search for the feature model. This permits a firm to have a centralized model, so that any updates to include new technologies or to remove any that have become outdated are immediately distributed to all its architects.

Once the feature model to be used has been obtained, the plug-in presents the architect with the decisions to be taken. The different decisions are presented to the architect as wizards pages. We opted for an interface of this kind to simplify the architect's task. While feature models are a widely used tool in the context of product lines, an architect specializing in multi-layer application development would not necessarily know this notation, so that its use would impose an additional burden.

VI. Related Work

In the area of architectural decision making, particularly stand out for their close relationship with our proposal two works of Zimmermann [10], [14]. They present a framework for the identification and modeling of recurring architectural decisions, and for converting those decisions into design guidelines for future development projects. In particular, Zimmerman proposes seven identification rules (IRs). These rules have their counterpart in our proposal. The main difference between our work and that of Zimmerman is the use made of those architectural decisions. In his work, the main objective is to gather information for use in future projects. Our focus is on using that information to simplify the process of obtaining an specific design of the application on which architectural decisions are made.

In the field of Web application development, Melia & Gomez [17] propose an extension to the model-driven methods of Web application development. Their proposal is closely related to the present work. Both pursue the same goal – to increase the architect's reliability when using model-driven techniques to design the architecture of a Web application. Nevertheless, their work focuses on RIA development, while ours is intended to encompass the entire class of multi-layer applications. Also, unlike our proposal, that of Melia & Gomez does not allow for control of the technologies used to implement the application, and neither does it provide any mechanism to log and store the decisions made by the architect for later use.

Finally, the studies of Antkiewicz [16] and Heydarnoori et al. [18] are of particular interest in the area of framework-based software development. Antkiewicz's techniques allow the modeling of specific designs for certain frameworks, and these models are then used to generate the source code. Heydarnoori et al. propose a technique for automatically extracting templates for implementing concepts of development frameworks. Unlike our work, the proposed techniques are very code centric, and their creation requires expertise in each framework employed. Our work is aimed at increasing the level of abstraction in the sense of being able to start from a technology-independent design, and progress to obtaining the final specific design by using the decisions made by the architect and model transformations.

VII. Conclusions and Future Work

This paper has addressed the problems facing the software architect when designing a multi-layer architecture. The complexity of these architectures, the complex relationship between functional and non-functional requirements and the high number of development frameworks and how they affect the non-functional requirements complicate the architect's task. We have proposed a technique for simplifying the architectural decision making process by means of the use of a feature model and a marked preliminary design. The proposed technique forms part of a broader procedure to address the transition from an initial design of an application to a design adapted to the architecture and technologies selected by the architect. This is a complex process that requires deep technical knowledge of the technologies involved. This complexity can be significantly mitigated by using model-driven development processes.

The next steps related to the architect's decision making will be based on improving the feature model's constraints. They can be used to incorporate additional information about quality attributes of the technologies, such as the performance of a given framework or the level of integration of two technologies. This additional information could be used to provide the architect with fuller and more precise initial suggestions.

Acknowledgments

This work was partially funded by the Spanish Ministry of Science and Innovation under Project TIN2012-34945, as well as by the Autonomous Government of Extremadura and FEDER funds.

References

[1] M. Fowler, *Patterns of Enterprise Application Architecture*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[2] R. Johnson, "J2ee development frameworks," *Computer*, vol. 38, no. 1, pp. 107 – 110, jan. 2005.

[3] T. C. Shan and W. W. Hua, "Taxonomy of java web application frameworks," *E-Business Engineering, IEEE International Conference on*, vol. 0, pp. 378–385, 2006.

[4] M. Raible, "Comparing jvm web frameworks," http://static.raibledesigns.com/repository/presentations/Comparing_JVM_Web_Frameworks_Jfokus2012.pdf, Jfokus, 2012.

[5] N. B. Harrison and P. Avgeriou, "How do architecture patterns and tactics interact? a model and annotation," *Journal of Systems and Software*, vol. 83, no. 10, pp. 1735–1758, 2010.

[6] L. Chung and J. C. S. do Prado Leite, "On non-functional requirements in software engineering," in *Conceptual Modeling: Foundations and Applications*, 2009, pp. 363–379.

[7] M. Dalgarno, "When good architecture goes bad," *Methods & Tools*, vol. 17, pp. 27–34, 2009.

[8] J. Garcia-Alonso, J. B. Olmeda, and J. M. Murillo, "Architectural variability management in multi-layer web applications through feature models," in *Proceedings of the 4th International Workshop on Feature-Oriented Software Development*, ser. FOSD '12. New York, NY, USA: ACM, 2012, pp. 29–36. [Online]. Available: http://doi.acm.org/10.1145/2377816.2377821

[9] J. Berrocal, J. García-Alonso, and J. M. Murillo, "Facilitating the selection of architectural patterns by means of a marked requirements model," in *ECSA*, ser. Lecture Notes in Computer Science, M. A. Babar and I. Gorton, Eds., vol. 6285. Springer, 2010, pp. 384–391.

[10] O. Zimmermann, "Architectural decisions as reusable design assets," *IEEE Software*, vol. 28, no. 1, pp. 64–69, 2011.

[11] R. Wojcik, F. Bachmann, L. Bass, P. Clements, P. Merson, R. Nord, and B. Wood, "Attribute-driven design (add), version 2.0," Software Engineering Institute, Tech. Rep. CMU/SEI-2006-TR-023, 2006.

[12] P. Avgeriou and U. Zdun, "Architectural patterns revisited - a pattern language," in *EuroPLoP*, A. Longshaw and U. Zdun, Eds. UVK - Universitaetsverlag Konstanz, 2005, pp. 431–470.

[13] K.-J. Stol, P. Avgeriou, and M. A. Babar, "Design and evaluation of a process for identifying architecture patterns in open source software," in *ECSA*, ser. Lecture Notes in Computer Science, I. Crnkovic, V. Gruhn, and M. Book, Eds., vol. 6903. Springer, 2011, pp. 147–163.

[14] O. Zimmermann, "Architectural decision identification in architectural patterns," in *WICSA/ECSA Companion Volume*, ser. ACM International Conference Proceeding Series, T. Männistö, M. A. Babar, C. E. Cuesta, and J. E. Savolainen, Eds., vol. 704. ACM, 2012, pp. 96–103.

[15] K. Czarnecki, S. Helsen, and U. W. Eisenecker, "Staged configuration through specialization and multilevel configuration of feature models," *Software Process: Improvement and Practice*, vol. 10, no. 2, pp. 143–169, 2005.

[16] M. Antkiewicz, K. Czarnecki, and M. Stephan, "Engineering of framework-specific modeling languages," *IEEE Trans. Software Eng.*, vol. 35, no. 6, pp. 795–824, 2009.

[17] S. Meliá, J. Gómez, S. Pérez, and O. Díaz, "Architectural and technological variability in rich internet applications," *IEEE Internet Computing*, vol. 14, no. 3, pp. 24–32, 2010.

[18] A. Heydarnoori, K. Czarnecki, and T. Tonelli Bartolomei, "Two studies of framework-usage templates extracted from dynamic traces," *IEEE Transactions on Software Engineering*, 2011.

CREATE: A Co-Modeling Approach for Scenario-based Requirements and Component-based Architectures

Marcel Ibe, Martin Vogel, Björn Schindler and Andreas Rausch
Technische Universität Clausthal
Clausthal-Zellerfeld, Germany
{marcel.ibe, m.vogel, bjoern.schindler, andreas.rausch}@tu-clausthal.de

***Abstract*—Requirements engineering and architectural design are key activities for successful development of software-intensive systems. Both activities are strongly intertwined and interrelated. Particularly, in early development stages requirements and architecture decisions are frequently changing. Thus, advanced systematic approaches are needed, which could minimize the risks of wrong early requirements and architectural decisions. The fundamental problem addressed in this paper is the development of inconsistencies at the advanced approaches for co-evolution of requirements and architectures. Inconsistencies lead to an incorrect consideration of requirements by the system under development and consequently to unfulfilled requirements. In this paper, a domain specific model-based approach is presented, which supports the co-evolution of requirements and architectures. The approach provides simplified scenario-based models for the description of requirements, which are suitable for validation by stakeholders. Furthermore, the approach provides a component-based model for a precise and complete description of architectures. Adequate inter-relations between scenario-based and component-based models are defined, which support the consistence maintenance.**

***Keywords*-requirements; architecture; evolution; consistency.**

I. INTRODUCTION

Requirements Engineering (RE) and Architectural Design (AD) are essential for successfully developing high-quality software-intensive systems. RE and AD activities are intertwined and iteratively performed [2]. The architecture of a software system must satisfy its requirements, yet architectural constraints might prohibit certain requirements to be realized. This might imply a change to the initial requirements or the selection of a different appropriate architecture. Further, additional requirements might be discovered during the development process, leading to changes in the architecture. Design decisions that are made early in this iterative process are the most crucial ones, because they are very hard and costly to change later in the development process.

In classical development processes, artifacts like, for instance, the requirements specification or the architecture are developed sequentially. This is also the case at iterative process models like the spiral life cycle model of Böhm [1]. The iterative, concurrent evolution of requirements and architectures demands that the development of an architecture is based on incomplete requirements. Also, certain requirements can only be understood after modeling or even partially implementing the system architecture. Nuseibeh [2] describes an advanced approach, which adapts the spiral life cycle model and aims at overcoming the often artificial separation of requirements specification and design by intertwining these activities in an interactive evolutionary software development process. This approach is called the twin peaks model. To map requirements into architectures and maintaining the consistency and traceability between the two Grünbacher et al. [3] introduces an intermediate model called Component Bus System Property (CBSP) (see Fig. 1). This model maps requirements to architecture elements by the CBSP model, which allows a systematic way to reconcile requirements with stakeholders.

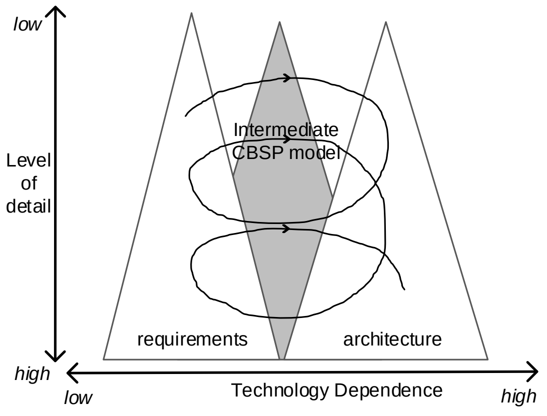


Figure 1. Intermediate model within the twin peaks [3]

Nevertheless, the advanced twin peaks model is kept very general. For instance, it does not specify the level of detail of requirements in relation to the architecture [4]. Due to the fact that there is no concrete advanced approach supporting the co-evolution of requirements and architecture we were commisioned by the German armed forces and the German government to undertake a research project. In order to be able to consider all required aspects, we made an expert survey. Therefore, we interviewed staff and leaders of three medium to big sized development projects with up to 30 project participants on customers and contractors side about their problems in the field of RE and AD.

A general mentioned problem was that the developed systems did not fulfill all requirements of the customers. The result of the survey was a list of the following reasons and derived guidelines:

- For the contractor the requirements were to informal, imprecise and incomplete. Requirements had to be repeatedly elicited and specified during architecture design. Hence, requirements on a software system should be complete and precise.
- At the elicitation process, the reconcilement of more precise and formal descriptions was to costly. The reason was the need of a detailed explanation by the contractor. For an improved reconcilement requirements descriptions should be precise as well as comprehensible. These guidelines are also mentioned by Nuseibeh [5]. Furthermore, the complexity of the models have to be manageable for validation by stakeholders.
- The most serious problem was caused by frequently changing requirements during architecture design. Changes frequently cause inconsistencies between requirements and architectures. Thus, requirements were frequently not fulfilled by the developed systems. Architectures have to describe how the system under development fulfills the given requirements by a precise definition of its structure and behavior. Consistency constraints between requirements and architectures should be defined, which enable an automatic support of the consistence maintenance.

Starting from this initial situation the target of the project was the development of a domain specific model-based approach, which fulfilles the mentioned guidelines. The subject of this paper is the presentation of the developed domain specific co-modeling approach CREATE for the description of requirements and architectures. Furthermore, the experiences at the application in practice are described. The presented approach is domain specific for interactive information systems like web-based systems and modern communication systems. In Section II, existing model-based approaches for the co-evolution of requirements and architectures are considered. In Section IV, the overall approach is introduced and in Section V the approach is shown at an example. Section VI contains a description of our experiences at the development and application of the approach in practice. Section VII includes a discussion of the results and pending points for future work.

II. Related Work

Existing model-based development approaches for requirements and architectures can be categorized into model-based approaches for requirements engineering, model-based approaches for architecture design and combined approaches.

Representative model-based approaches for requirements engineering are described in [6]–[8]. In [6], requirements are described by Unified Modeling Language (UML) [18] activity diagrams. A formal operational semantics enables execution of activity diagram specifications. The executed activity diagram specification serves as prototype for visualization of requirements. In the approach illustrated in [7], UML collaboration diagrams are enriched by user interface information in order to specify elicited requirements. These diagrams are transformed into complete dynamic specifications of user interface objects represented by state diagrams. These state diagrams are used for generation of prototypes. In [8], use case and user interface information are recorded at stakeholder interviews. Therefore, use case steps are enriched by scribbled dialog mockups. Prototypes are created, which visualize dialog mockups of use case steps in sequence for fast feedback of stakeholders. In general, these approaches have a well elaborated model structure for requirements engineering and improve the validation of requirements by stakeholders. On the other side, the mapping to the architecture is not precisely enough defined at these approaches to support a co-evolution of requirements and architectures.

Representative model-based approaches for architecture design are described in [9,10]. In Model-Driven Architecture (MDA) [9], the Computation Independent Model (CIM) can be used to describe business processes. The Platform Independent Model (PIM) may describe the structure and behavior of the software system. Component models like KobrA [10] are concrete approaches supporting MDA. In general, these approaches have a well elaborated model structure for architecture design and enable a detailed description of the structure and behavior of the software system. On the other side, these approaches do not support a co-evolution of requirements and architectures. The mapping between requirements and architectures is not precisely enough defined for this field of application.

Representative combined modeling approaches for requirements and architectures are described in [3,11,12]. In [11], a Requirements Definition Language (RDL) is used, which allows a structured definition of requirements. Meta model elements of the RDL are mapped to corresponding meta model elements of the Architecture Description Language (ADL). The approach described in [3] uses the intermediate model CBSP to map requirements to architecture elements. Different subtypes of CBSP elements allow classification of requirements. Requirements exhibit overlapping CBSP properties can be split and refined until no stakeholder conflicts exist. The Software Architecture Analysis Method (SAAM) [12] describes a method for a scenario-based analysis of software architectures. In this method, scenarios and architecture descriptions are developed iteratively. For each scenario it is determined whether a change of the architecture is required for execution. Based on the importance and conflicts of required changes an overall ranking of the developed scenarios is determined.

An advantage of these approaches is the combination of models for the description of requirements and architectures. On the other side, these approaches are very abstract and do not specify concrete models and mappings, which fulfill the conditions defined in the introduction for an adequate description of requirements and architectures.

Besides the stated existing approaches further approaches are conceivable, which are based on synthesis approaches [13] of complete state-based models from scenario-based models. Scenario-based and state-based models can potentially be used for the description of requirements and architectures. Consistency is, for instance, a subject of the approaches described in [14,15]. Unfortunately, these approaches are generally maintaining a complete consistency by means of a bijection. Architectures need to describe more details about the software system. These details have to be well separated from the requirements. Hence, an alternating correction of inconsistencies and not a bijection is required for the support of a co-evolution of requirements and architectures.

III. Contribution

The main contributions of this paper can be summarized as follows:

- Definition of a domain specific model-based approach for requirements engineering and architecture design in the sense of twin peaks. Requirements descriptions have to be precise and comprehensible. This necessitates a well-balanced trade-off between expressiveness and manageability of models for the description of requirements. Furthermore, the architecture has to provide a detailed description of the behavior and the resulting structure of the software system. In our domain specific approach, simplified scenario-based requirements models are defined for the description of requirements and component-based models for the description of architectures.
- Definition of consistency constraints which support a co-evolution of requirements and architectures. Complex dependencies between requirements and architectures cause a high complexity for consistence maintenance. Thus, not only for the requirements model, but also for the inter-relations between requirements and architecture models a well-balanced trade-off between expressiveness and manageability is necessary. In our approach, inter-relations between scenario-based requirements and component-based architecture models are provided, which enable an automatic consistence maintenance.
- This paper presents the results of the evaluation of the approach at real system development projects. Several iterations of phases for model definition and practice tests were required to find the presented solution. The approach is presented by a case study.

IV. Overall Approach

Our domain specific model-based approach supports concurrent development of requirements and architectures. An appropriate process for concurrent development is described by the twin peaks model [2]. In this model, requirements and architectures have an equal status and are evolved iteratively. This is illustrated by twin peaks (see Fig. 2).

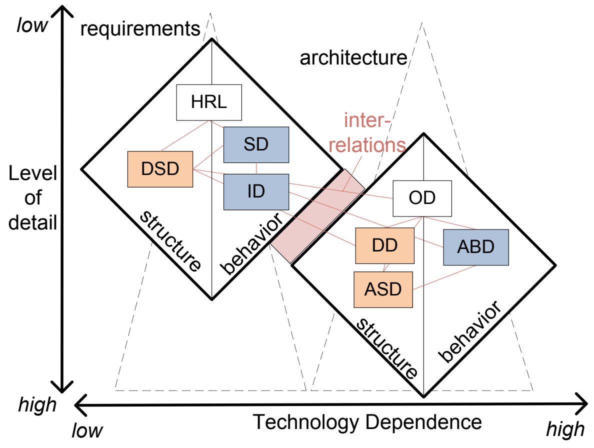


Figure 2. Co-modeling approach within twin peaks

Our domain specific model-based approach concretizes twin peaks by defining a concrete description technique. Diagrams are used for a precise description of requirements and architectures. These diagrams are illustrated within diamonds in the twin peaks model (see Fig. 2). The process flow of our approach begins with a formal description of inital requirements. Afterwards, the architecture is developed and consistence to the requirements is maintained continuously. Inconsistencies are resolved by changing requirements or the architecture.

The main contribution of our domain specific approach is the concrete description technique with well-defined inter-relations between requirements and architecture descriptions. It is well known that scenarios help to elicit and validate requirements [13]. A precise description of elicited requirements can be achieved by scenario-based models [13]. The co-modeling approach provides simplified scenario-based models for the description of requirements. Furthermore, the description is reduced to representative and concrete scenarios. Hence, the complexity of these models is manageable for the validation by stakeholders. The validation is improved by combining these models with models enabling visualization of requirements by user interface mockups [8]. During AD, the architecture of the co-modeling approach is developed. An architecture describes the behavior and the resulting structure of the software system precisely. This description is enabled by a component-model. Component-Based Software Engineering (CBSE)

[16] has been continuously improved and successfully applied over the past years. Systems are composed by existing software 'parts' called software components. Component models enable a precise description of component-based architectures [17].

In our domain specific model-based approach, diagrams are used to model structural or behavioral aspects of requirements or architectures. For instance, elicitation and specification of processes at the domain (e.g., business processes) is an important aspect at requirements engineering. In our approach, these processes can be described by a Scenario Diagram (SD). Thus, the SD is assigned to the behavior part of the requirements diamond (see Fig. 2). In Section V, the provided diagrams and their inter-relations are described in detail. Some models, for instance, the Hierarchical Requirements List (HRL) can be used to describe structural as well as behavioral aspects. Existing languages, such as UML [18], include among others structural and behavioral diagrams for the modeling of systems. Our domain specific approach uses exemplarily a subset of UML diagrams and their available model elements to formally describe requirements and architectures. Additional models are used to enable a visualization of requirements by user interface mockups.

Consistence maintenance during the development of requirements and architectures is supported by well-defined inter-relations between scenario-based requirements models and component-based architecture models (see Fig. 2). Inter-relations are also defined within these models. They are defined by associations between model elements and additional consistency constraints. The defined inter-relations enable an automatic consistence maintenance by, for instance, checking the consistency constraints and permitting changes not until detected inconsistencies are solved.

V. Modeling Example

Details of the description technique and the consistence constraints of the domain specific model-based approach are shown at a case study. The subject is the development of a library system.

A. Scenario Description

1) HRL: The HRL enables a text-based description of structural and behavioral requirements. They can be arranged hierarchically. In this way, it is possible to refine one requirement by several other requirements. In our example, the HRL contains some structural information about the system environment. The requirements list describes the system under development, the user of the system and an entity that should be managed by the system (red marked in Fig. 3 upper left). The requirement *show statistics* describes a desired behavior of the system. *Manage books* is a very general requirement and is refined by the requirement *show statistics*, which is more precise.

2) Domain Structure Diagram (DSD): The domain structure, e.g., the business structure, can be described by the DSD. It is based on UML Composition Structure Diagrams [18]. First, the domain structure consists of systems and persons as well as their ability to communicate to each other described by parts and connections of the DSD. The DSD *Library* describes the system to develop, the library system and a person, the employee (see Fig. 3 requirements left). The connection between the employee and the library system assumes that they can interact with each other. Furthermore, the DSD describes the relevant entities by parts. Currently, there is only one of the type *Book*. The parts of the domain structure (e.g., persons) have to be initially mentioned in the HRL (see gray line in Fig. 3).

3) SD: The description of processes at the domain (e.g., business processes) is an important task at requirements engineering. Processes can be described precisely by the SD, which is based on scenario-based UML Communication Diagrams [18]. SD describes representative scenarios at the domain, which have to be supported by the system under development. The scenarios are described as a sequence of messages between instances of the parts introduced in the DSD. Messages between two instances can only be sent if a connection exists between the corresponding parts of the DSD. In our example, the scenario *ShowBookStatistic* describes an interaction between an employee *m* and the library system with two messages (see gray line in Fig. 3 requirements upper right).

4) Interaction Mockup Diagram (ID): The ID can be used to visualize requirements to stakeholders. For this, it describes the messages of the SD by interaction mockups. Interaction mockups are user interface mockups, which visualize interactions of the system in general. In the SD, it is possible to describe the source and target of a message. In the ID, every message is detailed by exactly one interaction mockup for visualization. The scenario can be visualized to stakeholders by showing the interaction mockups step by step following the sequence of the messages of the SD. In our example, one interaction mockup shows the summary of all books of the message *1* in SD *ShowBookStatistic* (see gray line in Fig. 3 requirements). Another shows the detailed view of one book with its statistic.

B. Architecture Design

An architecture is a plan, which describes how a software system fulfills given requirements. The following architecture models allow a complete description of the behavior and the resulting structure of the system under development.

1) System Overview Diagram (OD): The OD of the architecture is based on the UML Use Case Diagram [18]. It is used to describe the most abstract structure and behavior of the system and its context by the system boundary and the associated use cases, which are called functions. The actors and the system context are derived from the DSD,

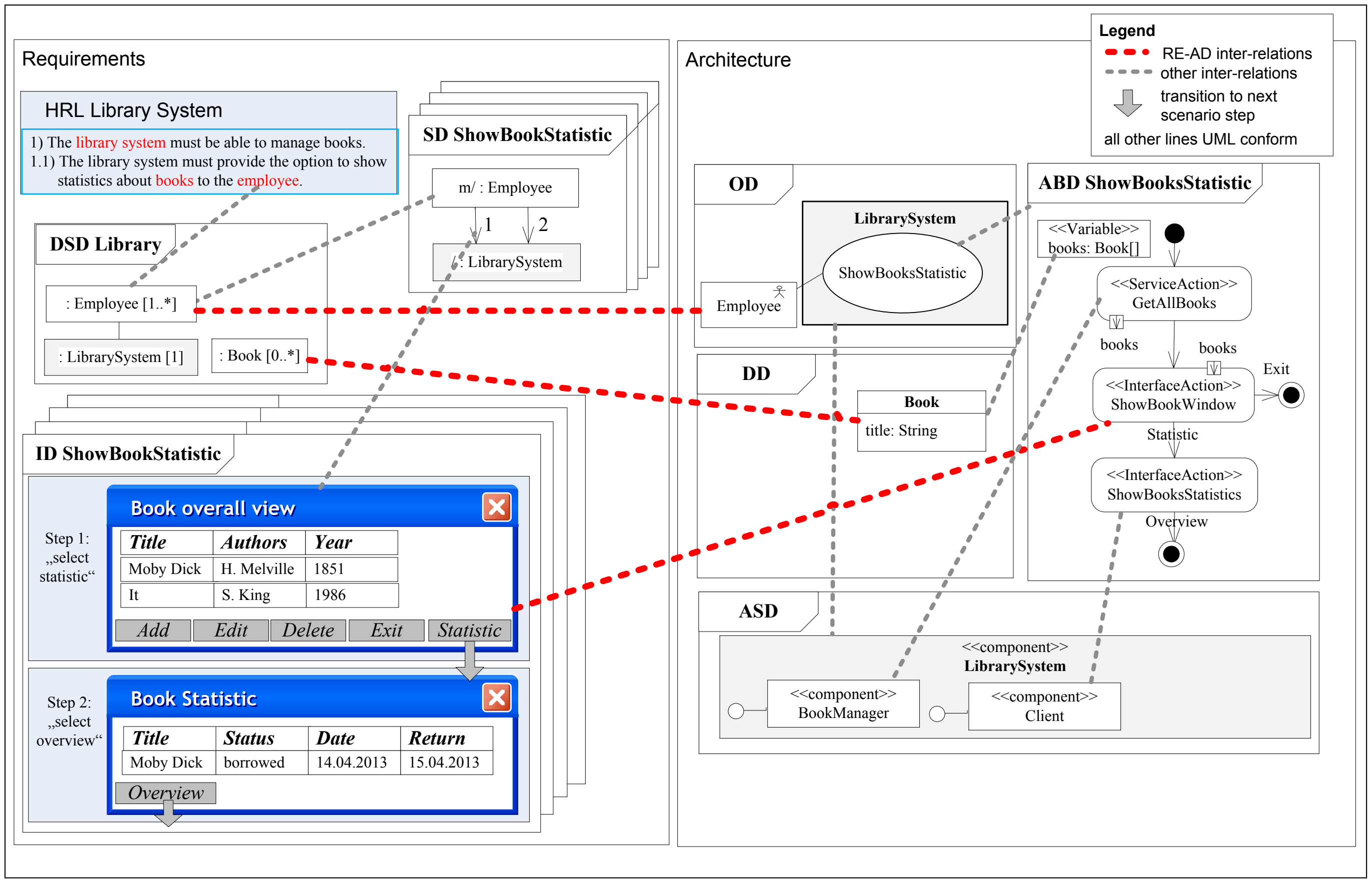


Figure 3. Requirements models, architecture models and inter-relations

the functions from SD. The OD of the example describes a system with the function *ShowBooksStatistics* and the actor employee (see Fig. 3 architecture upper right). Employee is connected to the function. This connection also exists between DSD (Employee) and SD (ShowBookStatistic). The process of a function is described by an architectural behavior diagram.

2) Architectural Behavior Diagram (ABD): The ABD describes the behavior of the software system and is based on the UML Activity Diagram including data flow. The ABD describes the process of the functions defined in OD completely. The function *ShowBookStatistic* is, for instance, described by the activity *ShowBookStatistic* (see gray line in Fig. 3 right). Within the ABD different action types like *InterfaceAction* and *ServiceAction* are used. A *ServiceAction* is performed by the system (e.g., a database call). An *InterfaceAction* describes an interaction of the system with its environment and is, therefore, associated with an interaction mockup of the ID. The action *ShowBooksStatistics* is, for instance, associated with an interaction mockup (see Fig. 3 right).

3) Data Diagram (DD): At a function described by ABD data objects can be used by the system. The DD is based on UML Class Diagrams [18] and describes the data types of the data objects. For example, the DD describes a type *book*, which is the type of the variable *books* of the ABD *ShowBooksStatistics* (see gray line in Fig. 3 architecture left). The data objects to be processed by the system are derived from the entities of DSD. The DD describes these entities in more detail. If there is a connection between two parts within the DSD, a relation must exist between the types of the parts and the corresponding data types in DD.

4) Architectural Structure Diagram (ASD): The ASD is based on UML Component Diagrams [18] and describes the internal components of the system under development and their offered interface as a black-box view. Subsequently, the components are further decomposed to refine their internal structure. The ASD *LibrarySystem* describes, for instance, the internal structure of *LibrarySystem* of the OD (see gray line in Fig. 3 architecture bottom). The internal structure is derived from the actions of the ABD. Hence, each component must be associated with an action of an ABD. The component *LibrarySystem* is refined by a component *BookManager*, which is associated with the action *GetAllBooks* of the ABD as well as the component *Client*.

C. Consistence constraints

The consistence maintenance is supported by the definition of inter-relations between requirements and architecture models. Inter-relations between models of the scenario-based requirements and the component-based architecture (e.g., between HRL and DSD) are described in Section V-A and V-B (see gray lines in Fig 3). Essential for the concurrent development are the inter-relations between requirements and architectures (see red lines in Fig 3). Inter-relations are defined in the term of associations and additional consistency conditions. A violation of a consistency condition means an inconsistency. We defined all necessary inter-relations. In the following, a subset of these inter-relations are exemplarily introduced, which is most suitable to explain the dependencies between our requirements and architecture descriptions:

- (1) The existence of an entity in the DD implies the existence of a corresponding type in the DSD.
- (2) The existence of a type in DSD whose part is directly connected with the part of the system to build implies the existence of a corresponding actor in the OD.
- (3) Every interaction mockup in the ID must be mapped on exactly one *InterfaceAction* in an ABD.

Probable changes during the development of concurrently evolved requirements and architectures are illustrated in Fig 4. While modeling the architecture it was noticed, that the system has not only to handle books. Also magazines should be managed. Hence, a new entity *Magazine* was added to the DD. As a consequence, the entity Media, as a generic term was introduced. Respectively two new inheritance relations were added. After this, the consistency condition (1) is violated. The DSD doesn't contain any corresponding element to these two new entities from the DD (arrows (1) in Fig. 4). A change in the requirements model was necessary, when it became clear that the manager needs other statistics about a book, then an employee. Hence, the manager as a new part of the system environment is added. In consequence, the condition (2) is violated. The manager is directly connected to the system but there is no corresponding actor in the OD (arrow (2) in Fig. 4). Because of the new needs of the manager, the scenario also has to be adapted. Depending on who uses the system, the shown information about a book varies. Thus, a new interaction mockup for the manager has to be added. This violates again the condition (3). The new interaction mockup is not mapped on an InterfaceAction from the ABD (arrow (3) in Fig. 4).

To correct these inconsistencies, a few further changes have to be made. It is necessary to add the entities *Media* and *Magazine* to the DSD. After this consistency condition (1) holds again (see arrows (1) in Fig. 5). To comply with the second condition, a new actor for the manager has to be introduced into the OD (arrow (2) in Fig. 5). Finally, a mapping from the added interaction mockup to an InterfaceAction is missing. One could map the new interaction mockup to an existing InterfaceAction or extend the ABD by a new InterfaceAction. By extending the ABD by the InterfaceAction *ManagerStats* the interaction mockup can be mapped on it (arrow (3) in Fig. 5). The new action may be processed by a new component *ManagerClient* at the ASD. By making these changes all consistency conditions were restored. As shown above, checking the consistency conditions helps to detect inconsistencies. An automatic support of the consistence maintenance can, for instance, be realized by permitting changes not until all inconsistencies are solved.

VI. Evaluation

The development of the co-modeling approach took place at research projects in cooperation with a public institution over a period of four years. At these research projects, we gave advice and supported to system development projects in order to test our results in practice. The goal of the overall approach is to support consistence maintenance of requirements and architectures in early development phases. The goal of the evaluation was to test the usability and the inconsistence prevention of our approach. At a first step, we developed the component-based architecture model for a precise description of the architecture. For reconcilement with stakeholders we developed a prototype generator, which is able to interpret the developed models. The stakeholders should validate the architecture and the consistence to their requirements with the aid of the prototypes. This approach was tested at a system development project over a period of one year. The subject of this project was a communication system. At this project, a model was developed comprising 20 system functions, 253 activity nodes and 35 data types. Conclusive it revealed that the usability of the approach has to be improved. The number of possible states described by the component-based architecture leads to less comprehensibility to stakeholders. They were not able to agree to the developed specifications. Consequently, the consistence maintenance of requirements and architectures could not be supported by this approach. Based on the results of this practice test we extended the approach by scenario-based models. This extended co-modeling approach, which is introduced in this paper, was tested in practice at a further system development project with a similar subject over a period of one year. In this period, the usability was significantly better. Stakeholders were able to agree to the visualized and scenario-based requirements. Furthermore, they were able to give helpful feedback, which leads to a big number of changes. We measured at three milestones the number of changes, the detected errors and especially remaining inconsistencies. Between these milestones we documented 500 changes and 67 errors. 8 of these errors were inconsistencies. The rate of inconsistencies to changes is low. For an indication, at a study described in [19], change

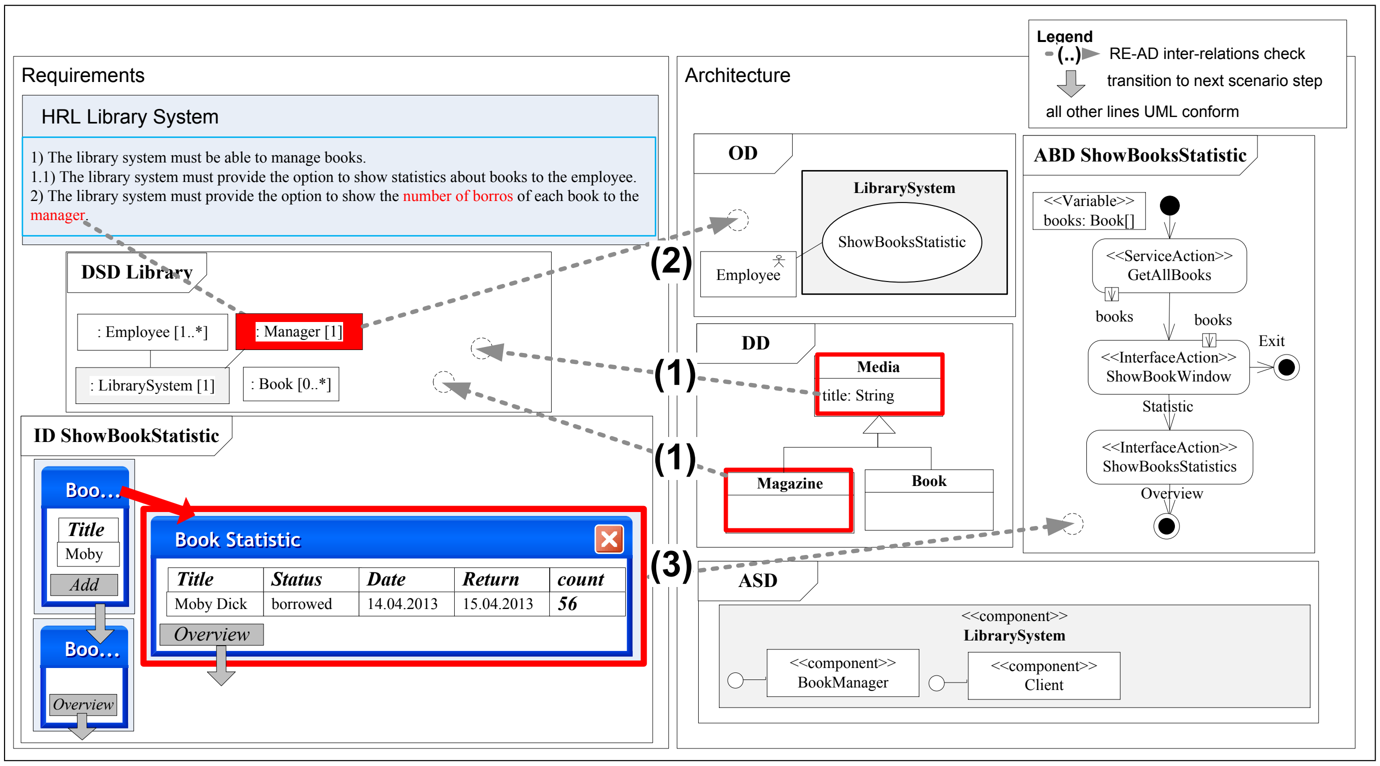

Legend
(..) RE-AD inter-relations check
transition to next scenario step
all other lines UML conform
Requirements
HRL Library System
1) The library system must be able to manage books.
1.1) The library system must provide the option to show statistics about books to the employee.
2) The library system must provide the option to show the number of borros of each book to the manager.
DSD Library
: Employee [1..*]
: Manager [1]
: LibrarySystem [1]
: Book [0..*]
ID ShowBookStatistic
Boo...
Title
Moby
Add
Book Statistic
Title
Status
Date
Return
count
Moby Dick
borrowed
14.04.2013
15.04.2013
56
Overview
Architecture
OD
LibrarySystem
ShowBooksStatistic
Employee
DD
Media
title: String
Magazine
Book
ASD
<<component>> LibrarySystem
<<component>> BookManager
<<component>> Client
ABD ShowBooksStatistic
<<Variable>> books: Book[]
<<ServiceAction>> GetAllBooks
books
<<InterfaceAction>> ShowBookWindow
Exit
Statistic
<<InterfaceAction>> ShowBooksStatistics
Overview
(1)
(2)
(3)

Figure 4. Changes at the requirements and architecture model

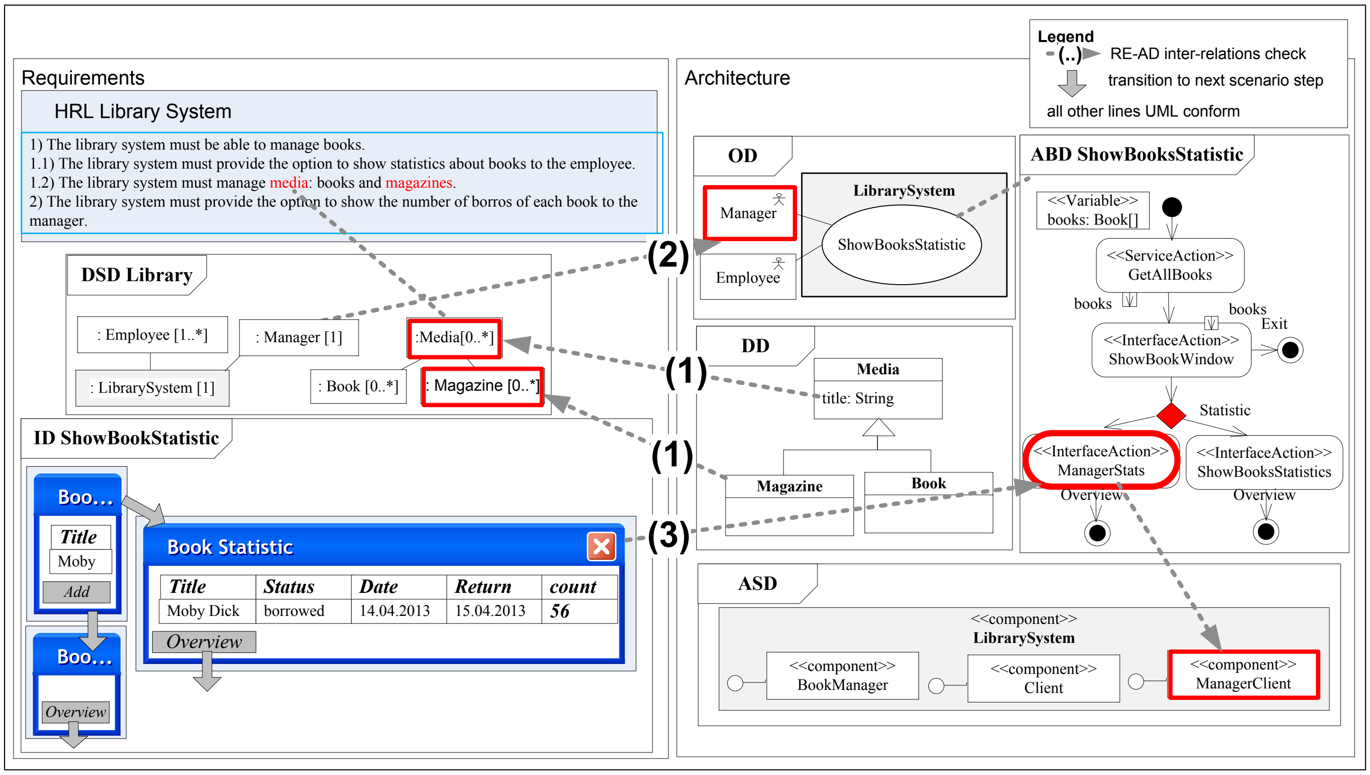

Legend
(..) RE-AD inter-relations check
transition to next scenario step
all other lines UML conform
Requirements
HRL Library System
1) The library system must be able to manage books.
1.1) The library system must provide the option to show statistics about books to the employee.
1.2) The library system must manage media: books and magazines.
2) The library system must provide the option to show the number of borros of each book to the manager.
DSD Library
: Employee [1..*]
: Manager [1]
:Media[0..*]
: LibrarySystem [1]
: Book [0..*]
: Magazine [0..*]
ID ShowBookStatistic
Boo...
Title
Moby
Add
Book Statistic
Title
Status
Date
Return
count
Moby Dick
borrowed
14.04.2013
15.04.2013
56
Overview
Architecture
OD
Manager
LibrarySystem
ShowBooksStatistic
Employee
DD
Media
title: String
Magazine
Book
ASD
<<component>> LibrarySystem
<<component>> BookManager
<<component>> Client
<<component>> ManagerClient
ABD ShowBooksStatistic
<<Variable>> books: Book[]
<<ServiceAction>> GetAllBooks
books
<<InterfaceAction>> ShowBookWindow
Exit
Statistic
<<InterfaceAction>> ManagerStats
<<InterfaceAction>> ShowBooksStatistics
Overview
(1)
(2)
(3)

Figure 5. Changes to solve the inconsistencies

data of requirements documents are analyzed. In this study, 88 changes, 79 errors, and 16 inconsistencies were detected.

VII. Conclusion and Future Work

The fundamental problem addressed in this paper was the development of inconsistencies at the advanced approaches for co-evolution of requirements and architectures. In this paper, a domain specific model-based approach was introduced, which supports a co-evolution of requirements and architectures. The approach uses a scenario-based model for a precise description of requirements and a component-based model for the description of architectures. Well-defined inter-relations enable an automatic consistence maintenance.

A frequently stated argument is the entailment of high costs for the development of precise requirements and architecture models at a software project. This can be countered by the fact that an incorrect consideration of requirements not uncommonly leads to complete project failures. Thus, maintaining the consistency at the co-evolution of requirements and architectures is important. Supporting this task by models enabling an automatic consistence maintenance reduces the risk of a project failure and costs for consistence maintenance. Furthermore, the developed models can be reused for automatic generation of code, test cases and documents like, for instance, requirements specifications. Nevertheless, the usage of formal models at a development project should, among others, be made conditional on the size of the project. At the beginning of a development project, the advantages and disadvantages of using formal models have to be weighed.

As future work, a further evaluation is planned to compare the effectivity of the co-modeling approach to other model-based approaches for requirements and architectures. Furthermore, it is planned to develop a tool for automatic consistence maintenance.

References

[1] B.W. Böhm, ”A spiral model of software development and enhancement”, IEEE Computer Society Press, vol. 21, May 1988, pp. 61–72.

[2] B. Nuseibeh, ”Weaving Together Requirements and Architectures”, IEEE Computer Society Press, vol. 34, March 2001, pp. 115–117.

[3] P. Grünbacher, A. Egyed, E. Egyed, and N. Medvidovic, ”Reconciling Software Requirements And Architectures With Intermediate Models” in Software and Systems Modeling. Springer, 2003, pp. 202–211.

[4] R. Ferrari and N. H. Madhavji, ”The Impact of Requirements Knowledge and Experience on Software Architecting: An Empirical Study” in Working IEEE/IFIP Conference on Software Architecture (WICSA), 2007, pp. 44–54.

[5] B. Nuseibeh and S. Easterbrook, ”Requirements Engineering: a roadmap” in Proceedings of the Conference on The Future of Software Engineering (ICSE), ACM Press, 2000, pp. 35–46.

[6] C. Knieke and U. Goltz, ”An executable semantics for UML 2 activity diagrams” in Proceedings of the International Workshop on Formalization of Modeling Languages (FML), ACM Press, 2010, pp. 3:1–3:5.

[7] M. Elkoutbi, ”Automated Prototyping of User Interfaces based on UML Scenarios” in Journal of Automated Software Engineering, vol. 13, Kluwer Academic Publishers, 2006, pp. 5–40.

[8] K. Schneider, ”Generating fast feedback in requirements elicitation” in Proceedings of the 13th international working conference on Requirements engineering: foundation for software quality (REFSQ), Springer-Verlag, 2007, pp. 160–174.

[9] A. G. Kleppe, J. Warmer, and W. Bast, ”MDA Explained: The Model Driven Architecture: Practice and Promise”, Addison-Wesley Longman Publishing Co. Inc., 2007

[10] C. Atkinson, J. Bayer, and D. Muthig, ”Component-Based Product Line Development: The KobrA Approach” in Software Product Line Conference, Denver, Kluwer Academic Publishers, 2000, pp. 289-309.

[11] R. Chitchyan, M. Pinto, A. Rashid, and L. Fuentes, ”COMPASS: Composition-Centric Mapping of Aspectual Requirements to Architecture” in Transactions on AspectOriented Software Development, 2007, pp. 3–53.

[12] R. Kazman, G. Abowd, L. Bass, and P. Clements, ”Scenario-Based Analysis of Software Architecture” in IEEE Softw., vol. 13, IEEE Computer Society Press, Nov. 1996, pp. 47–55.

[13] H. Liang, J. Dingel, and Z. Diskin, ”A comparative survey of scenario-based to state-based model synthesis approaches” in Proceedings of the 2006 international workshop on Scenarios and state machines: models, algorithms, and tools (SCESM), ACM Press, 2006, pp. 5–12.

[14] Y. Bontemps, P. Schobbens, and C. Löding, ”Synthesis of Open Reactive Systems from Scenario-Based Specifications” in Proceedings of Application of Concurrency to System Design, 2003, pp. 41–50.

[15] V. Garousi, L. Briand, C. Lionel, and Y. Labiche, ”Control Flow Analysis of UML 2.0 Sequence Diagrams” in Model Driven Architecture Foundations and Applications, 2005, pp. 160–174.

[16] C. Szyperski, ”Component Software: Beyond Object-Oriented Programming”, Addison-Wesley Longman Publishing Co. Inc., 2002.

[17] A. Rausch, R. Reussner, R. Mirandola, and F. Plasil, ”The Common Component Modeling Example: Comparing Software Component Models”, ser. Springer Lecture Notes in Computer Science, vol. 5153, 2008.

[18] OMG, ”UML, Version 2.2. OMG Specification Superstructure and Infrastructure”, 2009.

[19] V. R. Basili and D. M. Weiss, ”Evaluation of a software requirements document by analysis of change data” in Proceedings of the 5th international conference on software engineering, IEEE Press, 1981, pp. 314–323.

Reasoning About UML/OCL Models Using Constraint Logic Programming and MDA

Beatriz Pérez
Department of Mathematics and Computer Science,
University of La Rioja,
Logroño, Spain.
Email: beatriz.perez@unirioja.es

Ivan Porres
Department of Information Technologies,
Åbo Akademi University,
Turku, Finland
Email: ivan.porres@abo.fi

***Abstract*—The widespread adoption of Model Driven Engineering approaches has made of models to be cornerstone components in the software development process. This fact requires verifying such models' correctness to ensure the quality of the final product. In this context, the Unified Modeling Language (UML) and the Object Constraint Language (OCL) constitute two of the most commonly used modeling languages. We propose an overall framework to reason about UML/OCL models based on Constraint Logic programming (CLP). We use Formula as model finding and design space exploration tool. We show how to translate a UML model into a CLP program following a Meta–Object Facility (MOF) like framework. We enhance our proposal by identifying an expressive fragment of OCL, which guarantees finite satisfiability and we show its translation to Formula. We complete our approach by providing a Model Driven Architecture (MDA) based implementation of the UML to Formula translation. Our proposal can be used for software model design reasoning by verifying correctness properties and generating model instances of the modeled designs, using Formula.**

***Keywords*—UML; OCL; CLP; reasoning; verification**

I. Introduction

The widespread adoption of Model Driven Engineering (MDE) approaches has made of models to be cornerstone components in the software development process. This fact requires verifying both the completeness and correctness of such models to ensure the quality of the final product, reducing time to market and decreasing development costs. In this context, UML and OCL constitute two of the most commonly used modeling languages. The Unified Modeling Language (UML) [11] has been widely accepted as the de–facto standard for building object-oriented software. The Object Constraint Language (OCL) [10], on the other hand, has been introduced into UML as a logic-based sublanguage to express integrity constraints that UML diagrams cannot convey by themselves.

Unfortunately, in some occasions, possible design flaws are not detected until the later implementation stages, thus increasing the cost of the development process [4]. This situation requires a wide adoption of formal methods within the software engineering community. In this line, there have been remarkable efforts to formalize UML semantics to solve ambiguity and under specification detected in UML's semantics. The formalization and analysis of the specific UML modeled artifacts can be done by carrying out a semantic–preserving translation to another language [4]. The resulted translation can be used to reason about the software design by checking predefined correctness properties about the original model [4].

In this paper, we advocate for using the *Constraint Logic programming* (CLP) paradigm as a complementary method for UML modeling foundations, including models' satisfiability and inspection. More specifically, we focus on UML class diagrams (CD), annotated with OCL constraints, which are considered to be the mainstay of Object-Oriented analysis and design for representing the static structure of a system. Considering CD/OCL models as model representation, we propose an overall framework to reason about such models based on CLP. In particular, as model finding and design space exploration tool we use Formula [6], which stands on algebraic data types (ADT) and CLP, and which has been proved to provide several advantages, including more expressivity, over using other tools [7]. The defined framework is two–fold. Firstly, we have conceptually defined a proposal for the translation of CD/OCL models to Formula. Secondly, we have used a Model Driven Development (MDA) based approach [9] to automatically generate the Formula specification from a CD. As for the first contribution, we give a proposal for the translation of a UML model into a Constraint Satisfaction Problem following a multilevel Meta–Object Facility (MOF) like framework. We enhance our proposal by identifying a fragment of OCL which guarantees finite satisfiability, while being, at the same time, considerably expressive. We also show how to translate such OCL fragment to Formula, by giving, as an intermediate step, a representation of the OCL constraints as First-Order Logic (FOL) expressions. As for the second contribution, we use a model-to-text transformation tool to automatically transform a CD to Formula. Our framework can be used for software model design reasoning by checking correctness properties and generating model instances automatically using Formula, thus contributing to software designs' validation and verification.

The paper is structured as follows. Section II gives a brief introduction to Formula. An overview of our framework is presented in Section III. Section IV presents the translation of a CD to Formula, while Section V describes the chosen OCL fragment and its representation into Formula. The automatic MDA–based translation of a CD to Formula is presented in Section VI. Section VII summarizes the strengths and weaknesses of our approach and discusses related work. Finally, Section VIII covers our main conclusions and future work.

II. A Brief Overview of Formula

Formula distinguishes three units for modeling the problem: *domains*, *models* and *partial models*. A Formula *domain*

FD is the basic specification unit in Formula for an abstraction and allows specifying ADTs and a logic program describing properties of the abstraction. The logic programming paradigm provides a formal and declarative approach for specifying such abstractions [6], which in Formula are represented by *rules* and *queries*. A Formula *model FM* is a finite set of data type instances built from constructors of the associated domain *FD*, and which satisfies all its constraints [6]. Formula allows to specify individual concrete instances of the design-space or parts thereof, in a specific Formula unit called *partial model* [6]. A Formula *partial model FPM* is a set of instance-specific facts placed along with some explicitly mentioned unknowns, which correspond to the parts of the model *FM* that must be solved. *FPMs* allow unknowns to be combined with parts of the model that are already fixed [6].

```
1 domain MetaLevel extends UserDataTypes {
2 Star ::= {star}.
3 primitive Class ::= (name: String, isAbstract: Boolean).
4 primitive Association ::= (name:String,
          srcType:Class, srcLower:Natural, srcUpper:UpperBound,
          dstType:Class, dstLower:Natural, dstUpper:UpperBound).
5 Classifier ::= Class + Association.
6 errorBadMultInterval := Association(_,_, srcLower,srcUpper,_,_,_),
                                                 srcLower >srcUpper.
7 errorMetaDupAssoc := a1 is Association(name1,_,_,_,_,_,_),
     a2 is Association(name2,_,_,_,_,_,_),name1 = name2, a1 != a2.
8 ...
9 conforms:= !errorBadMultInterval & !errorMetaDupAssoc & ...}.
```

Figure. 1: An extract of a Formula domain.

Basically, a Formula *domain* consists of *abstract data types*, *rules* and *queries*. Firstly, *abstract data types* constitute the key syntactic elements of Formula. Based on the defined data types, a number of *rules* and *queries* are specified as logic program expressions, ensuring the remaining constraints [6]. Roughly speaking, *rules* specify implications and *queries* restrict the valid states by specifying forbidden states.

Abstract data types. They are defined by using the operator *::=*, indicating in the right hand side their properties by means of *fields*. A data type definition can be labeled with the `primitive` keyword, denoting that it can be used for the extension of other type definitions. Otherwise, the data type results in a *derived constructor*. As a way of example, in line 3 of Figure 1 we define the `Class` data type representing the UML *Class* meta–element constructor. The derived type `Classifier`, on the other hand, is defined as the union of the `Class` and `Association` types (see line 5 of Figure 1).

Around data types, Formula defines different categorizations of the structural elements as building blocks for defining Formula expressions. These elements are mainly Formula *terms* and *predicates*. As an example of a *term*, in line 7 of Figure 1 we show `Association(name1,_,_,_,_,_,_)`, which represents all instances of the `Association` term, where the first parameter is set to the `name1` property. The other fields of this type are filled with a do not-care symbol ('_'), so that Formula will find valid assignments. Terms are the basis for defining *predicates*, which constitute the basic units of data, used for defining *queries* and *rules*. An example of a predicate is `a1 is Association(name1,_,_,_,_,_,_)` (see line 7), where the variable `a1` is bound to the `Association` type.

Rules. Rules are specified by the operator `:-`, indicating, in the left hand, a simple term and, in the right hand, the list of *predicates* specifying the rule. A *rule* behaves like a universally quantified implication; whenever the relations on the right hand hold for some substitution of the variables, then the left hand holds for that same substitution [7]. The intuition of rules is *production*; they create new entries in the fact-base of Formula, populating previous defined types with facts representing the members in the collection presented in the rule.

Queries. Corresponding to rules where left hand side is a nullary construction [7]. A *query* behaves like a propositional variable that is true if and only if the right hand side of the definition is true for some substitution [7]. Queries are constructed by the operator `:=`, and can be also used like propositional variables to construct other queries. In particular, Formula defines in every domain the `conforms` standard query, where all constraints come together and which defines how a valid instance of the domain have to look like. Based on the *existential quantification* semantics of queries, the *universal quantification* can be achieved by verifying the negation of a query representing the opposite of the original predicate. For example, to ensure that upper bounds of associations' multiplicities are >= than lower bounds, we firstly need to define a query representing the existence of associations verifying the opposite (see line 6 of Figure 1). With this query, we are considering such incoherent situation as a valid state. Thus, to verify that such situation is invalid, we include the negation ('!') of the query in the `conforms` query (line 9).

Finally, to explore the design–space, Formula loads the specification of the domains and the partial models defined for the specific problem and executes the logic program. The execution finds all intermediate facts that can be derived from the given facts in the partial model, and tries to find valid assignments for the unknowns. This step is carried out by the *Formula solver*, which, in case it finds a solution that satisfies all encoded constraints, will reconstruct a complete instance model from this information made of known facts [6][7].

III. Encoding UML/OCL Models into Formula

As described previously, our proposal follows a MOF-like metamodeling approach, based on the framework the developers of the Formula tool give in [7]. Their framework provides a representation in Formula of part of the key concepts defined both at the MOF meta–level [11], representing the M2 level, and at the instance–level [11], representing the M1 level for the object diagram. The resulted Formula expressions are grouped in an only Formula *domain*, which is used by the *Formula solver* to find, if it exists, a valid set of instances of arbitrary class diagrams at the M1 level (conforming with their MOF meta–level representation) and its corresponding instances at the M0 level (conforming with their instance–level representation). We note that the authors in [7] do not give a specific approach for the translation of OCL constraints.

Based on this proposal, we have extended and modified it giving weight to four main aspects. Firstly, we have mainly focused on obtaining a more faithful representation of the MOF structural distribution, specifying a richer metamodeling framework. Our extended proposal is materialized into four different Formula units distributed along the MOF meta levels, which ease the application and the understandability of our approach, while promoting units reutilization. Secondly, we provide an approach based on the CLP paradigm for analyzing

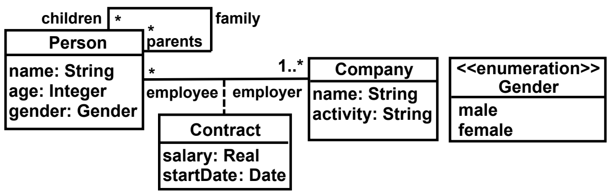


Figure. 2: Case study.

model instances of specific CDs, and not arbitrary ones as authors in [7] do, which we consider not enough when needed to reason about specific CDs. Thirdly, in contrast to [7], we give an approach for translating OCL constraints to Formula by; (1) identifying a significantly expressive fragment of OCL, and (2) providing its translation into Formula. Finally, we have implemented part of our translation approach based on MDA.

Each Formula unit defined in our approach contains two blocks of Formula expressions, related to the translation of the CD structural aspects (see Section IV) and its OCL constraints (see Section V), respectively. Our approach is illustrated with the case study of Figure 2, designed for explanation purposes covering basic aspects. It describes both the contractual relationship between a "Company" and a "Person", and the family recursive relationship connecting the class "Person".

IV. Translation of a CD's Structural Elements

This section presents a brief introduction of the rules we have defined to transform a class diagram ($\mathcal{CD}$), conforming with the UML metamodel [11] ($\mathcal{M}$), into Formula. Due to space reasons, in this paper we focus on a set of basic structural UML CD features (UML *class*, *attribute*, *association*) for being frequently used for modeling structural aspects of systems. Next, we briefly explain their translation classifying the generated Formula instructions into the different MOF levels. For the explanation, we lean on Table I.

Level M2. For each meta model element *Class*, *Association* or *Property* ϵ $\mathcal{M}$, we define a primitive Formula data type with the same name and with specific *fields* (see level M2 in Table I). For example, in the case of classes, we define the data type `Class(;)` ϵ *CPS*, with two `String` fields (`name` and `isAbstract`). The definition of these data types allows Formula to create Formula instances representing specific UML classes, associations and types of properties, respectively, at the M1 level. In the case of the *Property* element ϵ $\mathcal{M}$, we define a type for each *build-in type*, called *typeName*`Property`, with specific fields (see Table I). In addition to `Integer`, `String` and `Boolean`, included in [7], we also give support to `Real`, `LiteralNull` and `UnlimitedNatural` types. The data type `HasProperty(;)`ϵ *CPS* is also defined to represent the possession of a property by a classifier.

Level M1. Two groups of expressions are defined at this level. **[M1a.]** Each specific *class*, *association* and *property* ϵ $\mathcal{CD}$, is represented by a Formula instance of the corresponding constructor (`Class`, `Association` or `Property` ϵ *CPS* defined at level M2). By these Formula instances, we are explicitly representing, in contrast to [7], not arbitrary classes in a class diagram but specific ones. For example, the elements `ClassPerson` and `family` defined in `M1a` of Table I correspond to two Formula instances of the constructor `Class` and `Association`, respectively, defined at M2. In particular, specific properties ϵ $\mathcal{CD}$ are represented by a Formula instance of the corresponding `Property` constructor (e.g., *namePersonP is Str*`Property(...)` in `M1a` of Table I), and by an instance of the data type `HasProperty` ϵ *CPS*, representing the property's ownership (see Table I).

[M1b.] In order that Formula is able to generate instances of specific *class*, *association* and *property* ϵ $\mathcal{CD}$ to explore the concrete design–space, we need to create specific Formula data types representing each type of instance. For their definition, we have based on the description of the *Instances* package [11], in particular, on the *InstanceSpecification* element, for classes and associations, and on the *Slot* element, for properties. On the one hand, the definition of the UML *InstanceSpecification* element includes the classifier of the represented instance and the associated *InstanceValue* [11]. Taking this into account, for each *class c* ϵ $\mathcal{CD}$, we define a primitive Formula data type called `Instance`*c.name*`(;)`ϵ *CPS*, with two fields, representing the associated classifier and instance value, respectively (see level M1b in Table I). As a way of example, see the primitive data type `InstancePerson` in Table I. When the classifier is an association, the UML instance specification describes a *link* [11], so in this situations we name the created data types with the `Link` prefix. Since links connect class instances [11], for each *association a* ϵ $\mathcal{CD}$, we define a primitive Formula data type called `Link`*a.name*`(;;;)` ϵ *CPS*, which includes, additionally, the instance specifications of the associated classes (see for example `LinkFamily` in Table I). So that Formula can generate property's specific values, we define specific data types representing the property's slots, based on the specifications of the *Slot* element [11]. Taking this into account, for each *property* ϵ $\mathcal{CD}$, we define a primitive type called *p.name+p.owner.name*`Slot(;;)` ϵ *CPS* (e.g., `namePersonSlot` in Table I), which registers the owner, the property type and its value.

Level M0. Finally, in order that Formula can reason and search for valid instances of the specific classes, associations and properties of the source class diagram, we include the `Introduce(f,n)` command (used to add `n` terms of the element type `f`) with the corresponding `Instance`*c.name*, `Link`*a.name* or *p.name+p.owner.name*`Slot` data type, as `f`, and a specific number as `n`, to indicate the number of valid instances of such data type we want Formula to generate as part of the resulted object class diagram. For example, we define the `[Introduce(InstancePerson,2)]` command, so that Formula searches two valid instances of `InstancePerson` (see level M0 in Table I).

Finally, the Formula expressions resulted from the translation of a $\mathcal{CD}$ are grouped in four different Formula units. On the one hand, Formula expressions defined at the *meta–model level* (M2) are included into a Formula domain called $MetaLevel_{FD}$. Since the representation of the meta–level M2 is the same whatever $\mathcal{CD}$ is considered, this Formula domain is defined once and used for each $\mathcal{CD}$. An excerpt of the $MetaLevel_{FD}$ domain has been presented in Figure 1. On the other hand, Formula expressions defined at the *model level* (M1) are distributed into two different units; the $CDModel_{FM}$ model, which is constituted by the Formula expressions defined in `M1a`, conforming with the $MetaLevel_{FD}$ domain, and the $InstanceLevel_{FD}$ domain, constituted by the expressions defined in `M1b`. Finally, the Formula expressions at the *instance level* (M0) are included in the $CDInstance_{FPM}$ partial model.

TABLE I: Excerpt of the CD to Formula mapping.

Level		Class	Association	Property
M2		primitive Class ::= (name: String, isAbstract: Boolean).	primitive Association ::= (name: String, srcType: Class, srcLower: Natural, srcUpper: UpperBound, dstType: Class, dstLower: Natural, dstUpper: UpperBound).	primitive StrProperty::=(name:String, def:String, lower:Natural, upper:UpperBound). ... primitive LiteralNullProperty::=(name: String, def: Null,...). primitive UnlimitedNaturalProperty::=(name:String, def: UpperBound,.) Property::= StrProperty + ...+ userDataTypeProperties. primitive HasProperty ::= (owner: Classifier, prop: Property).
M1	a	*Translation pattern:* **Class***c.name* **is Class(***"c.name", c.isAbstract***)** *Example:* **ClassPerson is Class("Person", false)**	*Translation pattern:* *a.name* **is Association(***"a.name",* **Class("***a.memberEnd.at(1).type.name***"**, *a.memberEnd.at(1).type.isAbstract a.memberEnd.at(1).lowerValue, a.memberEnd.at(1).upperValue,* **Class("***a.memberEnd.at(2).type.name***"**, *a.memberEnd.at(2).type.isAbstract a.memberEnd.at(2).lowerValue, a.memberEnd.at(2).upperValue)* Example: **family is Association("family",Class("Person",false), 0, 2, Class("Person",false), 0, star)**	*Translation pattern:* *p.name+p.owner.name***P is** *p.type***Property***("p.name",p.default, p.lowerValue,p.upperValue)* **HasProperty(Class("**p.*owner,.name***"**, *p.owner.isAbstract),* *p.type***Property***("p.name",p.default, p.lowerValue,p.upperValue)***)** *Example:* **namePersonP is StrProperty("name","",1,1)** **HasProperty(Class("Person",false),StrProperty("name","",1,1))**
M1	b	*Translation pattern:* **primitive Instance***c.name* **::= (id: Integer, type: Class).** *Example:* **primitive InstancePerson::=(id: Integer, type: Class).**	*Translation pattern:* **primitive Link***a.name* **::=(id: Integer, type: Association,** a.memberEnd.at(1).name: **Instance**a.memberEnd.at(1).type.name, a.memberEnd.at(2).name: **Instance**a.memberEnd.at(2).type.name). Example: **primitive LinkFamily::=(id:Integer,type:Association, child:InstancePerson, parent:InstancePerson).**	*Translation pattern:* **primitive** *p.name+p.owner.name***Slot ::= (owner:Element, prop:***p.type***Property, value***: valueType***)** *Example:* **primitive namePersonSlot::= (owner: Element, prop: StrProperty, value:String).**
M0		*Formula instructions pattern:* **[Introduce(Instance***c.name, number* **)]** *Example:* **[Introduce(InstancePerson,2)]** *Example of the Formula generated instances:* **InstancePerson(93,Class("Person",false))** **InstancePerson(96,Class("Person",false))**	*Formula instructions pattern:* **[Introduce(Link**a.name, *number* **)]** *Example:* **[Introduce(LinkFamily,2)]** *Example of the Formula generated instances:* **LinkFamily(5, Association("family",Class("Person",false),0,2, Class("Person",false),0,star), InstancePerson(93, Class("Person",false)), InstancePerson(96, Class("Person",false)))**	*Formula instructions pattern:* **[Introduce(***p.name+p.owner.name***Slot**, *number* **)]** *Example:* **[Introduce(namePersonSlot,2)]** *Example of the Formula generated instances:* **namePersonSlot(InstancePerson(93,Class("Person",false)), StrProperty("name","",1,1),202)** **namePersonSlot(InstancePerson(96,Class("Person",false)), StrProperty("name","",1,1),201)**

Starting from these units, Formula can reason about the valid object class diagram, represented as instances of the elements of the *InstanceLevel*$_{FD}$ domain, conforming the given $\mathcal{CD}$, represented by means of the *CDModel*$_{FM}$ model.

V. TRANSLATION OF CLASS DIAGRAM'S CONSTRAINTS

OCL integrity constraints undecidability has been tackled in the literature by defining methods that allow UML/OCL reasoning at some level. Examples of such methods are [4], [13]; (1) those which allow only specific kinds of constraints, (2) those which consider restricted models, (3) methods which do not support automatic reasoning, or (4) those which ensure only semi–decidable models. Our approach, which would fit within the first type, identifies a significantly expressive fragment of OCL and provides its translation to Formula for OCL constraints' decidable reasoning. In this section, we show that our OCL fragment can be formally encoded in Formula, thus, we guarantee finite reasoning for every OCL CD's constraint expressed using the constructors of our OCL fragment. Next, we introduce the chosen OCL fragment and give the main idea of its translation to Formula. Due to space reasons, we translate a simple OCL constraint, which will serve as a reference explanation for the remainder elements of our OCL fragment.

Introduction to the chosen OCL fragment. We consider the OCL invariant `context C inv: expr(self)`, where `C` is the class ϵ $\mathcal{CD}$ to which the invariant is applied and `expr(self)`is an OCL expression resulting in a `Boolean` value for each `self` ϵ `C`. An OCL expression can be defined as a combination of *navigation paths* with OCL operations, which specify restrictions on those paths. A *navigation path* can be defined as a sequence of roles' names in associations (such as `p.children`, being `p` a `Person` instance in Figure 2), attributes' names (such as `c.name`, being `c` a `Company` instance in Figure 2), or operations (for example, `c.hireEmployee(p)`). Taking this into account, in Figure 3 we represent the syntax of our specific fragment, where `OCLExpr` is defined in a recursive manner. For example, an `OCLExpr` can be the result of applying relational operations to `AddExpr` expressions. Additionally, an `OCLExpr` can be the result of applying a boolean operation `BoolOper` to a `Path`, or a `Path` to which a `SelectExpr` is applied. An `OCLExpr` can be also constituted by boolean combinations of these OCL expressions (`not`, `and` and `or`). A `Path` expression represents the structural way of defining navigation paths, starting from a `PathItem`, by combining roles' names, attributes' names or operations, with the dot operator. For an explanation of OCL, we refer to [10].

```
OCLExpr     ≡RelExpr |Path BoolOper |Path SelectExpr
             not  OCLExpr | OCLExpr1 and OCLExpr2
             OCLExpr1 or OCLExpr2
Path        ≡PathItem | PathItem.Path
PathItem    ≡role | classAttr | operation
             roleName.role  | roleName.classAttr
             roleName.oper  | roleName.transClosuOper
RelExpr     ≡ AddExpr <,<=,>,>=,=,!= AddExpr
AddExpr     ≡ MulExpr | AddExpr +/- AddExpr
MulExpr     ≡ Path | MulExpr * Path | MulExpr/Path
SelectExpr  ≡ -> select( OCLExpr) BoolOp|
             -> select( OCLExpr) SelectExpr
BoolOper    ≡  -> size()| ->  forAll(OCLExpr)
```

Figure. 3: Syntax of the OCL fragment.

Our Translation Approach. Formula does not have a concept similar to that of OCL invariants but gives the possibility of defining queries, which provide a way to represent invariant semantics. As way of example of our approach, in this section we introduce the basic rule for translating OCL invariants where the `OCLExpr` corresponds to a simple relational expression `RelExpr`. We explain this rule by applying it to the invariant `context Person inv: self.age >=18`, which formalizes the constraint "The people working on a company must be older than 18 years old" (see Table II).

First–step. This step is carried out by means of an interpretation function *FOL*(), which translates each OCL expression `expr(self)` defined in an instance `self` ϵ `C`, into a First–Order Logic (FOL) formula defined in the variable `self` (see label (1) in the first step of Table II). Basis in first order logic states that the universal quantifier corresponds to a negated existential, so the previous expression is equivalent to the one label (1'), where `FOL(not expr(self))`, corresponds to the mapping of `not expr(self)` into First–Order Logic (FOL).

Second–step. Each constraint logic program P can be translated into FOL according to its *Clark Completion* P^* [8].

TABLE II: Translation of an invariant and example of use.

Translation of a `RelExpr` invariant	
OCL Invariant: `context C inv: expr(self)`	
First–step:	`∀self ∈ C` *FOL*`(expr(self)).` (1) `¬(∃self ∈ C` *FOL*`(not expr(self)).` (1’)
Second–step:	`¬(`*FOL**`(C)` *FOL**`[`*FOL*`(not expr(self))])` (2)
Third–step:	`query:=`*CLP*`(`*FOL**`[`*FOL*`(not expr(self))])` `conforms := ! query.` (3)
Example of application	
OCL Invariant: `context Person inv: self.age >=18`	
First–step:	`∀self ∈ Person age(self)>=18.` (1) `¬(∃self ∈ Person age(self)<18).` (1’)
Second–step:	`¬(∃ ageSlot(self,def,val) val<18).`(2)
Third–step:	`query:=ageSlot(self,_,val), val<18.` `conforms := ! query.` (3)

Roughly speaking, the *Clark Completion* of an atom or predicate symbol can be represented as a combination of term expressions and rules, evaluated in variables, giving a `true` result. The inverse translation, that is, from the FOL representation of *P* (*P**) to *P*, can be carried out by applying inverse versions of the *Clark Completion* algorithm [3], which compile specifications into the logic program it directly specifies. Taking this into account, the second step is devoted to represent the semantics given by the affirmative evaluation of *FOL*`(not expr(self))` in the collection of instances `self ∈ C`, by means of Formula expressions. Since paths in OCL are defined in terms of instances of the class diagram, and in our approach such instances are defined by means of the data types defined in the $CDInstance_{FPM}$ partial model, such Formula expressions are written in terms of the `Instance`*className*, `Link`*associationName* and/or *propertyName+ownerName*`Slot` data types. Based on this premise, in this second step we rewrite the FOL expression *FOL*`(not expr(self))` in terms of Formula expressions by applying a second function called *FOL**(). This function *FOL**() basically represents the predicate *FOL*`(not expr(self))` by using the corresponding Formula terms and predicate symbols $\in InstanceLevel_{FD}$, and Formula constraints, in such a way that the resulted expression is evaluated to `true` (see step labeled (2) in Table II). In particular, the application of this step to our constraint consists of representing `age(self)<18` in terms of the `ageSlot` whose `val` property is less than 18.

Third–step. Taking into account the semantics of queries in Formula, the FOL expression given in the second step is finally represented by means of the definition of a `query` and the verification of its negation in the `conforms` query (see step labeled (3) in Table II). This step is materialized by means of the application of the function *CLP*(), which basically rewrites the terms resulted from (2), and joins them by ‘,’.

Thus, the translation of an invariant is carried out by means of the composition of the three defined functions. Next, we make some remarks regarding the translation of the remainder elements in our OCL fragment (see Table III). In particular, excluding the `select` and `transitive closure` elements, whose translation requires extra attention, we consider that the translation of the remainder OCL elements can be easily understood by considering our previous explanations.

Select operation. Since this operation refers to obtaining a subcollection from a set of elements, its translation consists of defining a new Formula data type and populate it with the facts representing the members in the collection we want to select (see the first and second lines, respectively, of the translation of the `select` operation in Table III). As a way of example, if we want to collect the female employees of a company, we define the type: `FemaleEmp::= (self: InstanceCompany, sele:InstanceEmployee),` and populate it by means of the rule: `FemaleEmp(self,sele) :- LinkContract(_,_,sele,self),genderPSlot(sele,_,val), val=female.`, which gathers only female employees.

TABLE III: Translation of part of our OCL fragment.

OCL expression	Translation approach
`E1 and E2`	*CLP*`(`*FOL**`(`*FOL*`(E1)))&`*CLP*`(`*FOL**`(`*FOL*`(E2)))`
`E1 or E2`	*CLP*`(`*FOL**`(`*FOL*`(E1)))|`*CLP*`(`*FOL**`(`*FOL*`(E2)))`
`not E`	*CLP*`(`*FOL**`(`*FOL*`(not E)))`
`C-> size()`	`count(`*CLP*`(`*FOL**`(`*FOL*`(C)))).`

`C->`
`forAll(c|exp(c)`
	`query:=`*CLP*`(`*FOL**`(`*FOL*`(not exp(c)))).`
	`conforms:= ! query.`

`C->`
`select(c|exp(c)`
	$\mathtt{S}_{C,expr}$`Type::=(self:`$\mathtt{T}_{self}$`,sele:`$\mathtt{T}_{sele}$`)`
	$\mathtt{S}_{C,expr}$`Type(self,sele):-`
	CLP`(`*FOL**`(`*FOL*`(exp(c))))`

Transitive closure. Transitive closure is normally needed to represent model properties which are defined in a recursively fashion. The translation of closures is not straightforward since they are not finitely axiomatizable in first order logic, and OCL also does not support them natively [2]. Nevertheless, it is possible to define the transitive closure of relations which are known to be finite and acyclic. In particular, for its translation we have based on both, the definition of transitive closure provided in [2], and the representation in CLP of acyclicity constraints provided in [7] (page 3), and proposed a translation based on defining Formula rules, considering the fact that CLP exposes fixpoint operators via recursive rules. Additionally, the translation of this operation allows us to support *aggregation*.

Finally, the Formula model resulted from the translation of a CD model annotated with OCL constraints (that is, the 4 Formula units including the Formula translation of the OCL constraints), is used by Formula for reasoning about it. More specifically, the tool inspects the Formula model looking for a valid and non–empty instantiation of the CD/OCL model to proof its satisfiability. If the result is empty, the defined CD/OCL model is not satisfiable. Otherwise, Formula proposes a conforming instantiation model of the defined CD/OCL model, according to the desired software system settings.

VI. Automatic Translation

In order to manually transform a CD into the Formula language, a professional with both UML and Formula skills may be required. Also, such an encoding process may entail a big effort depending on the CD used. The challenge is to perform such a transformation in a viable and cost–effective way. The complexity of some software designed models together with their possibility of change over time, make the manual transformation of every CD into the input language of a model finder tool a cumbersome and costly endeavor. To overcome these challenges, we use an MDA tool-approach, which allows us to automatically carry out the transformation from the CD to Formula. Among the large amount of MDA-based tools in the literature, we are interested in those with support for customizable model–to–text transformations. The idea is to define only one set of transformations for all CDs. Finally, we have chosen the MOFScript Eclipse plug-in [5], which we have already used in previous works [12]. In our particular case, we use the UML 2.0 metamodel and the specific CD as the model, defined using the UML2 Eclipse plug-in [5]. As far as the Formula program generation is concerned, we have defined several MOFScript transformation scripts that generate the different Formula units with the translation of the structural CD elements. In particular, we have defined a set of MOFScript transformation rules, grouped into different MOFScript files, employed to produce the print Formula structures that constitute the three Formula units in our approach, which depend on the specific CD.

VII. Discussion and Related Work

As described previously, the formalization and analysis of UML CDs can be done by means of translating the model to other language that preserves its semantics, and finally, using the resulted translation to reason about the design. Taking into account that there is not an only language for materializing such translation, and that several translation approaches can be established using a same language, a discussion about the semantic support of the language, together with the strengths and weaknesses of the particular translation approach, is worthwhile. Our work bets on using Formula for the semantics preserving translation of the models to be verified. As for the use of Formula instead of other analyzers, in particular, Formula authors present in [7] a comparison with other tools, both SAT (Boolean Satisfiability) solvers and alternatives such as *ECLiPSE* and *UMLtoCSP*, focusing mainly on Alloy [1], for being the closest tool to Formula. Although the Formula authors provide a careful comparison with Alloy in [7], it is worth noting the strengths of Formula, such as a more expressive language or its model finding problems, which are in general undecidable.

Our approach follows a multilevel MOF-like framework based on the one proposed in [7]. On the one hand, we propose a more faithful representation of the basic UML metamodel and instance domain elements [11]. We consider that providing a translation which captures the structural distribution of the MOF architecture can contribute to ease the application and understandability of the representation of a CD/OCL model into Formula. We also give support for the translation of more metamodel elements (such as full support to generalization, property types other than *Integer*, *String* and *Boolean*, including user defined data types, property's multiplicities, etc.), thus providing a richer framework. Additionally, we enhance the proposal given in [7] by identifying an expressive fragment of OCL, which guarantees finite satisfiability and providing a formalization of the transformations from such OCL fragment to Formula. At this respect, several related works can be cited, being one of the most complete proposals the one given in [13]. In [13] the authors define a fragment of OCL called OCL–lite, and prove the encoding of such a fragment in the description logic $\mathcal{ALCI}$, so that Description Logic techniques and tools can be used to reason about CD annotated with OCL–lite constraints. A difference of this approach with ours is the fact that, although the chosen fragment is quite similar than ours, we have tried to identify a simplest fragment so that no element included in it can be inferred from other constructors in the fragment by applying direct OCL equivalences (such as the `implies` operator). In our particular case, there are several OCL operations and expressions whose representation in Formula is straightforward by applying equivalences (such as the `exists`, `isEmpty/notEmpty`, `xor`, or `reject`). Finally, there are other operations (such as `oclIsTypeOf`, considered in [13]) that can not be represented into Formula, but we give support to other not straightforward operators, such as *transitive closure*, not normally included in related works.

VIII. Conclusion and Future Work

We present an overall framework to reason about UML/OCL models based on the CLP paradigm, using Formula. Our framework provides a way to translate a UML model into Formula, following a MOF-like approach. We also identify an expressive fragment of OCL, which guarantees finite satisfiability and we provide an approach for translating it to Formula. Our proposal can be used for model design reasoning by verifying correctness properties and generating model instances automatically using Formula. We provide an implementation of our CD to Formula proposal, being the implementation of the OCL fragment a remaining work.

Acknowledgments

This work has been partially supported by the Academy of Finland, the Spanish Ministry of Science and Innovation (project TIN2009-13584), and the University of La Rioja (project PROFAI13/13).

References

[1] K. Anastasakis and B. Bordbar and G. Georg and I. Ray, "UML2Alloy: A Challenging Model Transformation," Proc. of the 10th International Conference on Model Driven Engineering Languages and Systems (MoDELS'07), LNCS, vol. 4735, 2007, pp. 436-450.

[2] T. Baar, "The Definition of Transitive Closure with OCL - Limitations and Applications," Proc. of the 5th Andrei Ershov International Conference in Perspectives of System Informatics (PSI'03), LNCS, vol. 2890, 2003, pp. 358-365.

[3] A. Bundy, "Tutorial Notes: Reasoning about Logic Programs," Proc. of the 2nd International Logic Programming Summer School (LPSS'92), LNCS, vol. 636, 1992, pp. 252-277.

[4] J. Cabot and R. Clarisó and D. Riera, "Verification of UML/OCL Class Diagrams using Constraint Programming," Proc. of the 2008 IEEE International Conference on Software Testing Verification and Validation Workshop (ICSTW'08), IEEE Computer Society, 2008, pp. 73–80.

[5] Eclipse Modeling Project, Available at: http://www.eclipse.org/modeling/. Last visited on August 2013.

[6] Formula - Modeling Foundations, Website: http://research.microsoft.com/en-us/projects/formula. Last visited on August 2013.

[7] E. K. Jackson and T. Levendovszky and D. Balasubramanian, "Automatically reasoning about metamodeling," Software & Systems Modeling, February, 2013, doi:10.1007/s10270-013-0315-y.

[8] J. Jaffar and M. J. Maher and K. Marriott and P. J. Stuckey, "The Semantics of Constraint Logic Programs," J. Log. Program., vol. 37, 1998, pp, 1-46.

[9] OMG, OMG Model Driven Architecture, Document omg/2003-06-01, 2003, Available at: http://www.omg.org/. Last visited on August 2013.

[10] OMG, Object Constraint Language OCL, OMG Specification, Version 2.2, 2010, OMG Document Number: formal/2010-02-01. Available at: http://www.omg.org/spec/OCL/2.2. Last visited on August 2013.

[11] OMG, UML 2.4.1 Superstructure Specification, Document formal/2011-08-06. Available at: http://www.omg.org/. Last visited on August 2013.

[12] B. Pérez and I. Porres, "Authoring and Verification of Clinical Guidelines: a Model Driven Approach", Journal of Biomedical Informatics, vol. 43, num. 4, 2010, pp. 520-536.

[13] A. Queralt and A. Artale and D. Calvanese and E. Teniente, "OCL-Lite: A Decidable (Yet Expressive) Fragment of OCL*," Proc. of the 25th International Workshop on Description Logics (DL'12), Description Logics, vol. 846, 2012, pp. 312-322.

Weaving Crosscutting Concerns into Inter-process Communications (IPC) in *AspectJ*

Ali Raza, Dr. Stephen W. Clyde
Computer Science Department
Utah State University
Logan, Utah, USA
ali.raza@aggiemail.usu.edu

***Abstract*—Implementing crosscutting concerns for message-based *inter-process communications (*IPC*)* are difficult, even using *aspect-oriented programming* languages (AOPL*)* such as *AspectJ*. Many of these challenges are because the context of communication-related crosscutting concerns is typically a conversation consisting of message sends and receives. Other challenges stem from the wide variety of IPC mechanisms, their inherent characteristics, and the many ways in which they can be implemented, even using a common communication framework. Additionally, current AOPL do not provide pointcuts for weaving of advice into high-level IPC abstractions like conversations. This paper describes an extension to *AspectJ*, called *CommJ*, with which developers can implement communication-related concerns in cohesive and loosely coupled aspects.**

Keywords-modularity; aspect-oriented programming (AOPL); crosscutting concerns; AspectJ; software reuse and maintenance.

I. Introduction

*Inter-process communications (*IPC*)* are ubiquitous in today's software systems, yet they are rarely treated as first-class programming concepts. Instead, developers typically have to implement communication protocols indirectly using primitive operations, such as *connect*, *send*, *receive*, and *close*. The sequencing and timing of these primitive operations can be relatively complex. For example, consider a distributed system that uses the *Passive File Transfer Protocol* (PFTP) to move large data sets from a client to a server. The server would enable communications by listening for connection requests on a published port, e.g., 21. A client would then initiate a conversation, i.e., an instance of the PFTP protocol, with a connection request to the server on that port. Figure 1 shows a typical sequence of messages following the initial connection request.

Neither the client's nor the server's side of the conversation is simple. In fact, to ensure responsiveness for end users and to handle multiple simultaneous clients, both the client and server might execute parts of a single conversation on different threads, making it even harder to follow concurrent conversations dynamically. A system using PFTP could be further complicated by communication-related requirements not central to primary objective of moving large amounts of data, such as logging, detecting network failures, monitoring

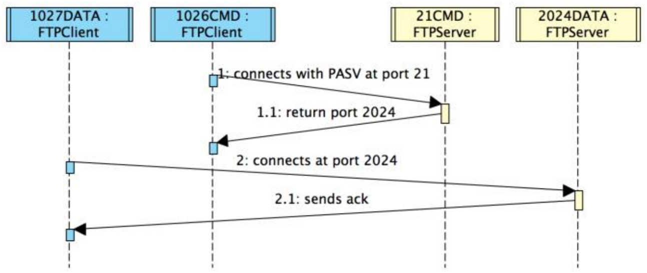


Figure 1: PassiveFTP interaction Diagram

congestion, and balancing load across redundant servers.

From a communications perspective, these concerns (and many others not listed above) are what *Aspect-oriented Software Development* (AOSD) refers to as *crosscutting concerns*, because they pertain to or *cut through* multiple parts of core or base concepts. Directly implementing these concerns in a typical system can cause the *scattering* and *tangling* of code. *Scattering* occurs when the same or very similar logic exists in multiple places in the software. *Tangling* occurs when single software component is complicated by logic for secondary concerns. Scattering and tangling often occur together.

AOSD, which first started to appear in the literature in 1997 [12, 25], reduces scattering and tangling of code by encapsulating crosscutting concerns in first-class programming constructions, called *aspects* [15]. In strongly typed languages, an *aspect* is an *Abstract Data Type* (ADT) with all of the same capabilities as an object class. However, an aspect can also contain *advice* methods that encapsulate logic for addressing crosscutting concerns and *pointcuts* for describing where and when the advice needs to be executed. More specifically, a pointcut identifies a set of *join points*, which are temporal places in the execution of the system for where and when *weaving* of advice takes place [15].

AspectJ is an AOPL that extends *Java* for aspects [14-17]. It allows programmers to *weave* advice into join points that correspond to constructor calls or executions, methods calls or executions, class attribute references, and exceptions.

It is possible for skilled software developers to create reusable, well-encapsulated crosscutting concerns in a traditional *object-oriented programming language* OOPL.

However, the difference between AOPLs and\an OOPLs is that AOPLs offer convenient mechanisms for separating crosscutting concerns from core functionality and for following a principle called *obliviousness* [18]. Although perhaps poorly named, *obliviousness* is the idea that core functionality should not have to know about crosscutting concerns [13].

The problem is that *AspectJ*, like other AOPLs, does not support the weaving of advice into high-level communication abstractions, such as conversations. Our work, called *CommJ*, extends *AspectJ* so developers can weave crosscutting concerns into IPC in a modular and reusable way, while keeping the core functionality oblivious to those concerns. See Section II for a high-level overview. Section III describes a conceptual model that provides a theoretical foundation for *CommJ*, namely its message event joint points (see Section IV) and event tracking (see Section V). Section VI describes base aspects central to *CommJ's* implementation. To validate *CommJ*, we have created a library of reusable aspects for common communication crosscutting concerns and have applied them to a variety of sample systems (see Section VII). Then, Section VIII discusses how application programmers can write their own communication aspects. Related work is presented in Section IX. Finally, Section X summarizes the current state of *CommJ* and outlines our future work.

II. HIGH-LEVEL OVERVIEW

Overall *CommJ* enables the partitioning of a complex communication problem into manageable cohesive concepts and promotes greater reuse with better maintainability. Figure 2 shows an architectural block diagram that represents relevant conceptual layers and their dependencies. The following paragraphs describe the high-level components and their dependencies.

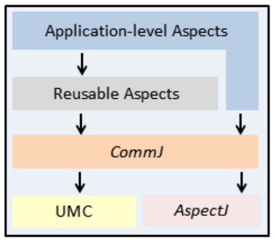


Figure 2: *CommJ* Architectural Block Diagram

In general, a universe model is a formal description of a closed universe of things, as well as their relationships, properties, interactions, and behaviors. Figure 3 shows part of our universe model for IPC, which we refer to as the UMC or *Universe Model of Communication*. Section III describes a portion of UMC in more detail.

CommJ is an *AspectJ* library that implements message-event join points and keeps track of conversations. A software developer that wants to use communication-related aspects simply has to include this library. Sections IV - VI explain how *CommJ* implements the join points, keeps track of conversations, and base abstractions for the application programmers, respectively.

The *Reusable Aspect Library* (RAL) is a toolkit-like collection of communication aspects that application programmers should find useful for in many different kinds of applications. They include aspects for measuring turn-around times, tracing conversations, and introducing behaviors into complex, multi-step protocols, like PFTP. Section VII describes this library in more detail.

Application-level Aspects are those written by the application programmers, either by using the abstractions provided by *CommJ* or by specializing the aspects in RAL. Section VIII discusses how these application-level aspects can encapsulate complex crosscutting behaviors in an understandable and maintainable way, without sacrificing obliviousness or flexibility.

III. UNIVERSE MODEL FOR COMMUNICATIONS

The UMC establishes a conceptual framework for discussing and reasoning about network-based communications. Figure 3 shows a portion of this model, namely the part that deals with message concepts. The full UMC includes other concepts, like connections, that we do not discuss here for brevity.

The central idea of the portion presented in Figure 3 is that of a *Message Event*, which is the “happening” of a message being sent (i.e., *Sent Event)* or a message being received (i.e., *Received Event*). It is a time point related to a particular message and is part of a *Conversation* following a *Protocol*. Every *Received Event* must have a corresponding *Message Received* object, which is simply a message in the role of having been received. Similarly, every *Sent Event* must have a *Message Sent* object. Also, consistent with theoretical foundations for IPC [28], all the *Message Events* in a system form a partial ordering; the events on a single thread are totally ordered; and a message's *Sent Event* always comes before its *Received Event(s)*.

Message in Figure 3 is an abstraction that represents data sent from one process to another as part of conversation. Each *Message* can be associated with at most one send and possible many receive events, which is the case for multicasts or broadcasts. The *Message* class contains abstract reflection methods for retrieving *message identifying information* (MII*)*, which consists of message, conversation, and protocol identifiers. Application developers implement these methods for their specific types of messages and then *CommJ* uses those implementations in keeping track of conversations. Since these methods are abstract and are implemented in the application, developers remain in full control of their message structure.

Even though the UMC focuses on communications, it includes *Channel*, *Thread*, *Node*, and *Process* classes to help provide context information for the individual messages and conversations

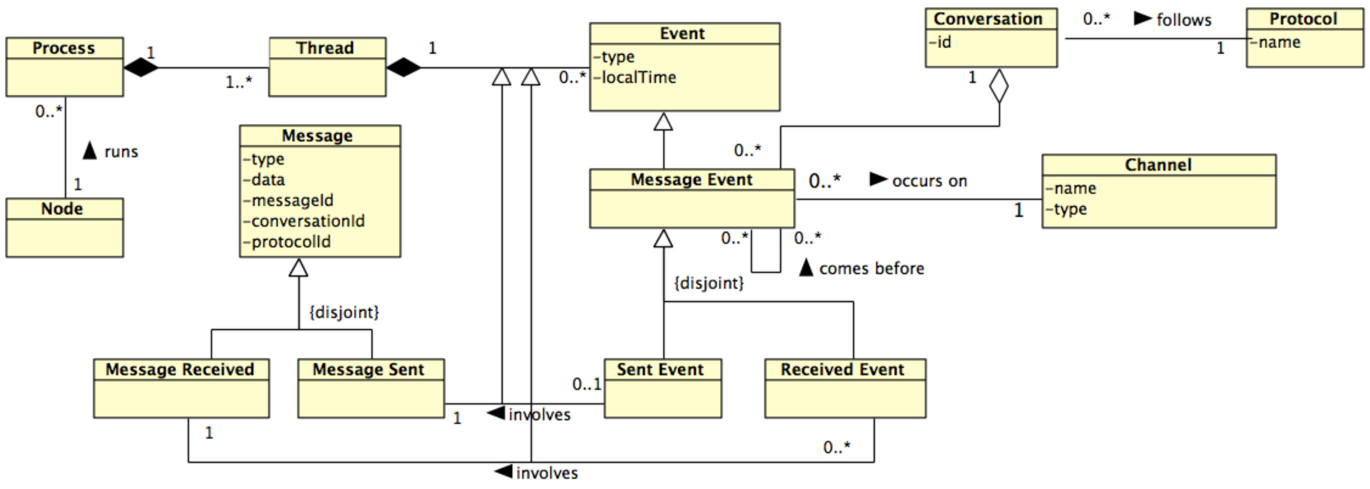


Figure 3: A conceptual model for UMC

IV. Message Event Join Points

Communication join points fall into two general categories: message related and connection related. Since this paper is focusing on *Message Events*, we only discuss the former here.

As mentioned earlier, join points represent places and times where/when advice can be executed. In *AspectJ*, they correspond to constructors, methods, attributes, and exceptions. Advice can be executed before, after, or around these various *contexts*. *CommJ* adds conversations to the list of possible *contexts*, but unlike the advice contexts in *AspectJ*, a conversation is not tied to a single programming language construct. Instead, a context *in CommJ* can be either:

A - an entire conversation from a process's perspective (see Figure 4)

B - any sequence of message send or receive events in the conversation as seen by a process

C - a single send or receive event in a conversation

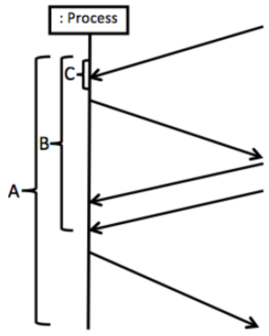


Figure 4: Conversations in *CommJ*

The green boxes in Figure 5 are *CommJ* classes that implement join points for these different kinds of contexts. *MultiStepConversationJP* represents join points for entire conversations, as well as joints points for sequences of events within a conversation. *RRConversationJP* (i.e., request-request conversation join points) also represents join points for complete conversations, but only those that follow request-reply protocols. *MultiStepConversationJP* could be used for the same, but *RRConversationJP* includes optimizations for this common type of conversation. *SendEventJP* and *ReceiveEventJP* implement joint points for individual message events.

A developer can implement crosscutting concerns, define conversation-related pointcuts, and weave advice into any of above join points by specializing the corresponding abstract *CommJ* aspects, shown in yellow in Figure 5.

V. Event Trackers and Registries

Behind the scenes, *CommJ* uses *JoinPointTrackers*, which are *monitors* [22] that perform pattern matching on communication events, to track individual events and to organize them into high-level conversation contexts. Since the monitoring of communications is itself a crosscutting concern, *JoinPointTrackers* are implemented as aspects that weave the necessary monitoring logic into places where communication event may take place. Although *CommJ* can support many different kinds of *JoinPointTrackers*, Figure 5 only shows one special kind of tracker, namely *MessageJoinPointTracker*, which has been specifically designed for send and receive events on standard JDK sockets and channels.

When a *MessageJoinPointTracker* discovers a relevant communication event, it creates a join point instance, e.g., a *SendEventJP*, correlates it with other events in the same conversation, and then adds it to a registry, namely the *MessageJPRegistry* shown in Figure 5. Advice in a communication aspect can access these join point objects to obtain context information, like the conversation's start time, channel, or the protocol.

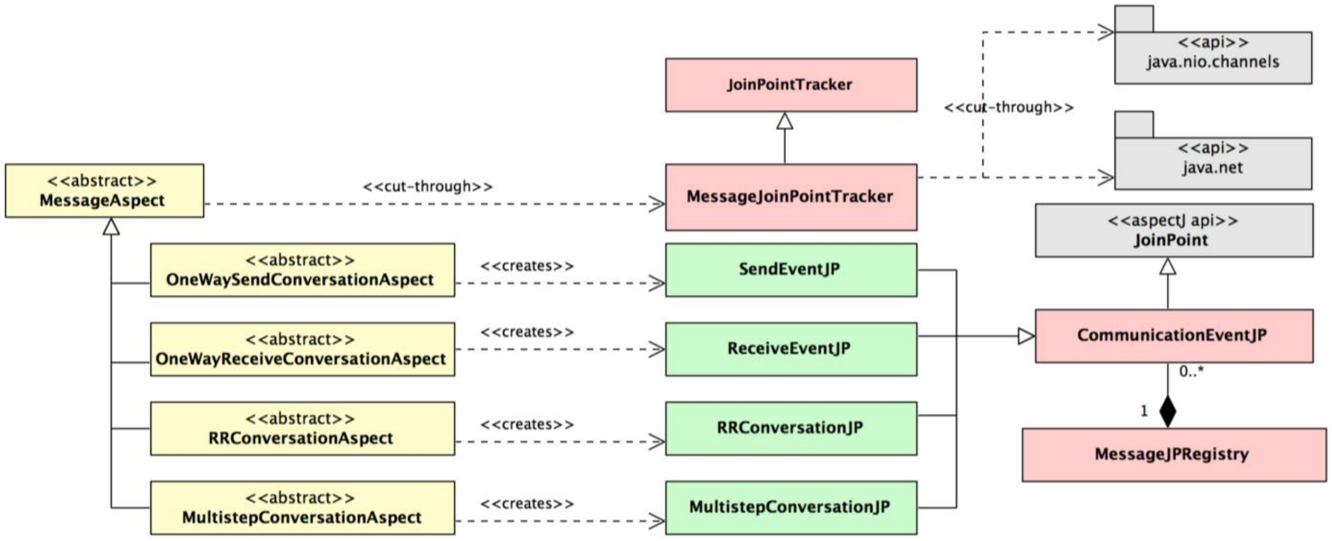


Figure 5: *CommJ* Message Event Join Points and Aspects

VI. BASE ASPECTS

All communication aspects are ultimately derived from abstract *MessageAspect*, which provides concrete pointcuts that dynamically track send and receive events (See Figure 6 for more details). For space considerations, the full definitions of the pointcuts are not shown, and are not necessary for understanding their purpose. However, it is important to note that they take join point objects as parameters, because this is how advice based on these pointcuts can access communication contexts.

```
public abstract aspect MessageAspect{
   public pointcut MessageSend(SendEventJP jp) ...
   public pointcut MessageRecieve(ReceiveEventJP jp) ...
}
```

Figure 6: Pointcuts in *MessageAspect*

The four specializations of *MessageAspect* in Figure 5 correspond to four different kinds of conversation contexts, as mentioned earlier, and extend *MessageAspect* with pointcut abstractions that are meaningful to those contexts (see Figures 7a-7d). Developers can create their own application-level communication aspects that inherit from these aspects and include advice based on these pointcuts.

The *OneWaySendAspect* is relatively trivial because it represents a simple one-message conversation from the message sender's perspective. Similarly, the *OneWay-ReceiveAspect* represents a one-message conversation from the message receiver's perspective.

The *RRConversationAspect* extends *MessageAspect* with pointcuts for conversation beginnings and conversation ends. Developers can use this aspect to weave advice before, after, or around simple request-reply conversations, either from a conservation initiator or responder perspective.

The *MultistepConversationApsect* is the most complex of the four. In addition to pointcuts for conversation beginnings and ends, it provides a way for applications to specify arbitrarily complex communication protocols, which define the message patterns that comprise conversations. A process can participate in a conversation with one or more *ProcessRoles*. See Figure 8.

The key to working with complex protocols is that an aspect developer can formally define them in terms of *ProcessRoles* and then *ProcessRoles* in terms of finite state machines (see *State Machine* in Figure 9.) For example, consider a communication protocol that involves three processes, *A*, *B*, and *C*, and where A starts a conversation by sending a message to *B* and waits for a response. When *A* receives a response *B*, it then sends a message to *C* and waits for a response. When *A* receives a

```
public abstract aspect OneWaySendAspect
                        extends MessageAspect{
   public pointcut ConversationBegin(SendEventJP jp)....
}
```

Figure 7(a): OneWaySend aspect in RAL

```
public abstract aspect OneWayReceiveAspect
                        extends MessageAspect{
   public pointcut ConversationEnd(ReceiveEventJP jp)....
}
```

Figure 7(b): OneWayReceive aspect in RAL

```
public abstract aspect RRConversationAspect
                        extends MessageAspect{
   public pointcut ConversationBegin(RRConversationJP jp) ....
   public pointcut ConversationEnd(RRConversationJP jp) ....
            ....
}
```

Figure 7(c): RRConversation aspect in RAL

```
public  abstract aspect MultistepConversationAspect
                        extends MessageAspect{
  public pointcut ConversationBegin(MultistepConversationJP jp)....
  public pointcut ConversationEnd(MultistepConversationJP jp)....
   ....
}
```

Figure 7(d): MultistepConversation aspect in RAL

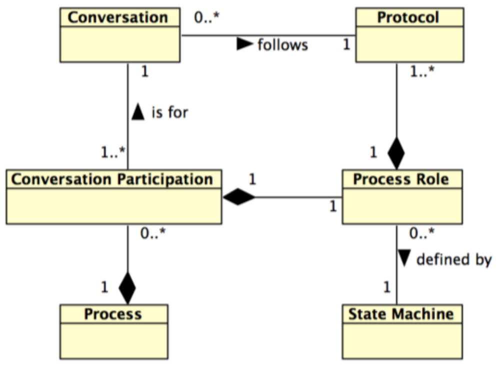


Figure 8: Process participation in conversations by roles and role

response from *C* it sends a final message to both *B* and *C*. Figure 9 shows a finite state machine for the *A ProcessRole* of this protocol. The *B* and *C ProcessRoles* are considerably simpler and are not shown here.

The *CommJ StateMachine* class includes a *buildTransitions* method that allows developers to define state machines in terms of states and message-event transitions. Figure 10 shows the implementation of this method to define a *StateMachine* for the sample *ProcessRole* shown in Figure 9.

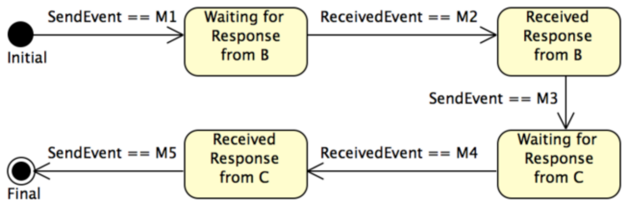


Figure 9: Sample Process Role

```
public class SampleProcessRole extends StateMachine{
  ....
 @Override
 public void buildTransitions(){
  addTransition("Initial", 'S', "M1", "WaitingRspFromB");
  addTransition("WaitingRspFromB ", 'R', "M2", " ReceivedRspFromB");
  addTransition("ReceivedRspFromB", 'S', "M3", " WaitingRspFromC");
  addTransition("WaitingRspFromC",  'R', "M4"," ReceivedRspFromC");
  addTransition("ReceivedRspFromC",  'S', "M5"," Final");
 }
            ....
}
```

Figure 10: State Machine configuration for sample Process Role

VII. Reusable Aspects Library

Aspects in the RAL are also derived from the base aspects in *CommJ*. They represent general crosscutting concerns commonly found in applications with significant communication requirements. Table 1 lists some of the aspects currently in the RAL and Figure 11 shows part of the implementation of first one, *TotalTurnAroundTime-Monitor*. Note how the advise in this aspect follows the *Template Method* pattern [29]. This allows developers to quickly adapt it to the specific needs of their application by overriding the *Begin* and *End* methods. Other aspects in the RAL make use of this and other reuse techniques to easily integrate them into existing or new applications.

We expect that RAL will continue to grow as new generally applicable communication aspects are discovered, implemented, and documented.

```
public aspect TotalTurnAroundTimeMonitor
          extends MultistepConversationAspect{
   private long startTime = 0;
   private long turnAroundTime = 0;
   before(MultistepConversationJP jp):
ConversationBegin(jp){
         startTime = System.currentTimeMillis();
         Begin(jp);
   }
   after(MultistepConversationJP jp): ConversationEnd(jp){
         long turnaroundTime = (System.currentTimeMillis() –
                  startTime)/1000;
         End(multiStepJP);
   }
   public getTurnAroundTime { return turnAroundTime; }
  protected void Begin(MultistepConversationJP jp){
      // Specialization of this aspect should override the
method
  }
  protected void End(MultistepConversationJP jp){
      // Specialization of this aspect should override the
method
  }
  ...
}
```

Figure 11: A code snippet of *TurnAroundTimeAspect*

VIII. Application-Level Communication Aspects

As mentioned, aspect developers implement and add application-level aspects into core application logic by either reusing RAL aspects or specializing the base aspects in *CommJ*. As an example, this section describes the implementation of an application-level aspect that weaves performance measurements in the multistep protocol,

TABLE I. Six of the aspects in the RAL and their Descriptions

Aspect Name	Description
TotalTurnAroundTimeMonitor	Provides virtual helper methods for conversations which help programmers to override RAL aspects in their applications
MessageLoggingByConversation	Log messages by conversations in a developer-defined format and repository
MessageEncryption	Add session-level encryption/decryption to communication protocols
NetworkNoiseSimulator	Allows developers to add noise, message log, and message duplication to network communications, which is useful for system testing
NetworkLoadBalancer	Helps programmers balance message loads across two more communication channels
VersionControlAspect	Helps programmers manage multiple version of messages structures for their applications

introduced in the previous section. For discussion purposes, assume that the performance measurements are a rolling window of throughput and average-conversation turn-around time statistics. Also, assume that the core application considers a unit of work to be the completion of a conversation that follows this protocol. So, we can measure throughput for a unit of time, say 1 minute, by simply counting the number of these conversations completed in that minute. The average turn-around time is the average of timespans from conversation start times to conversation end times. The rolling window keeps track of these statistics for the current minute and 10 previous minutes. Figure 12 in the next page shows the key snippets of an aspect that implement this performance measure crosscutting concern.

First notice how this advice is derived from *TotalTurn-AroundTimeAspect* and in doing so, it can reuse its implementation of the conversation turnaround time

```
public aspect MyAppPerformanceMonitor
          extends TotalTurnAroundTimeMonitor{

  private Stats[] statsList = new ArrayList[11];
  private int currentStatsIndex = 0;

  @Override
  protected void End(MultistepConversationJP jp) {
   // Get number of elapsed minutes since beginning of current
Stats
   long elapsedMinutes = Min(Stats[currentStatsIndex].
getMinutesSinceStartTime(), 10);
   // Roll Stats window forward, if necessary
   for (int i=0; i<elapsedMinutes; i++){
    currentStatsIndex++;
    if (currentStatsIndex>10)
      currentStatsIndex=0;
      Stats[currentStatsIndex].Reset();
   }
   currentStats.addCompleteConversation(getTurnaroundTime);
  }
}

class Stats{
  private long startTime;
  private int completeConvCount;
  private double avgTurnaroundTime;

  public Stats{
    Reset();
  }

  public Reset(){
    startTime = currentTime;
    completeConvCount = 0;
    avgTurnaroundTime = 0;
  }
  public long getMinutesSinceStartTime() {
    // using current time, compute and return the number of
minutes since the start time of this Stats object. A zero means
we still in the same minute
  }

  public void addCompleteConversation(double
newTurnaroundTime) {
    avgTurnaroundTime =
((completeConvCount*avgTurnaroundTime) +
newTurnaroundTime)/(++completedConvCount);
  }
}
```

Figure 12: performance measure crosscutting concern

concept directly. Then, it adds the *Stats* array for holding the rolling window of statistics and some additional behavior to the ending of a conversation to compute the statistics.

IX. Related Work

We found many papers that talk about using aspect-oriented technology for communication-related cross-cutting concerns, such as replication [5], persistence [9], synchronization [8, 16], and remote pointcuts [6]. To date, we have not found any other work that extends the possible contexts and join points for aspects to conversations or sequences of events in a specific conversation. The closest idea discusses the composition of communication abstractions by separating out definition of communications from the definition of other aspects [7]. Although this work is of value, we believe that *CommJ* enables better modularity while preserving obliviousness.

Marco, et al., describe a Java-based communication middleware, called *AspectJRMI*, that applies AOPL concepts to modular design and the implementation of *RMIs* [27]. Their primary contribution is the decomposability of *RMI* into small crosscutting concerns.

Other related ideas deal with the definition of reusable communication constructs in languages, like *Erlang*, which is based on processes communicating via asynchronous message passing [26, 21]. However, these approaches do not inherently encourage the separation of crosscutting concerns from core application requirements.

Gary, et al., describe an approach for building customized protocols using *Cactus* – a system in which micro-protocols are implementing individual attributes of transport [1]. More complex protocols can then be composed from these micro-protocols. Dirk, et al., show how to separate the definition of communication from the definition of other system functionality [2]. A paper on extensible client-server software by Coady, et al., talks about requiring a clear separation of core services from those that should be customizable [3]. Remi, et al., talk about concurrent event-based AOPL and define an approach of writing concurrent aspects [11]. All these works address research objectives different from *CommJ* and only indirectly related to our research.

X. Summary and Future Work

This paper introduced the notation of communication aspects and discussed an *AspectJ* framework, i.e., *CommJ*, for weaving aspects into inter-process communications. It then describes the design and implementation of some of *CommJ* key components, namely the base aspects. It also provides an overview of a toolkit that consists of reusable communication aspects and doubles as a proof of concept, since these aspects can be directly applied to a wide range of existing applications.

Based on preliminary evidence, we believe that *CommJ* is capable of encapsulating a wide range of communication-related crosscutting concerns in modular aspects. However, more research and experimental evidence is needed. We plan to conduct real world experiments using *CommJ* to verify its benefits in software reuse and maintenance. We also hope to gather more empirical evidence of *CommJ* value by increasing the number of aspects in the RAL and by continuing to expand the number and types of applications that use *CommJ*.

Those interested in trying out *CommJ* or contributing to it can obtain a copy of the framework from http://commj.cs.usu.edu.

REFERENCES

[1] Wong G., Matti A. and Richard D., "A Configurable and Extensible Transport Protocol," IEEE/ACM Transactions on Networking, Vol 15, No 6, 2007.

[2] Heuzeroth D., Lowe W., Ludwig A., and Amann U., "Aspect-Oriented Configuration and Adaptation of Component Communication," Proceedings of the Third International Conference on Generative and Component-Based Software Engineering GCSE 01.

[3] Coady Y., et al., "Can AOP Supports extensibility in Client-Server Architectures," In Proceedings, ECOOP Aspect-Oriented Programming Workshop 2001.

[4] Bergmans L., Tekinerdogan B., Glandrup M. and Aksit M., "Composing Software from Multiple Concerns: A Model and Composition Anomalies," ICSE00.

[5] Nishizawa M., Chiba S., "Jarcler: Aspect Oriented Middleware for Distributed Software in *Java*," Research Report Computer Science Department, Tokyo Institute of Technology (2002).

[6] Nishizawa M., Chiba S., and Tatsubori M., "Remote Pointcut – A Language Contruct for Distributed AOP," AOSD 2004.

[7] Daniel L., et al., "Explicitly distributed AOP using AWED," AOSD 2006.

[8] Carlos A., Sobral L., and Miguel P., "Reusable Aspect-Oriented Implementations of Concurrency Patterns and Mechanisms," AOSD06.

[9] Soares S., Laureano E., and Borba P., "Implementing Distribution and Persistence Aspects with *AspectJ*," OOPSLA 2002.

[10] Antunes M., et al., "Separating Replication from Distributed Communication: Problems and Solutions," International Conference on Distributed Computing Systems Workshop, 2001.

[11] Douence R., Botlan D., Noye J., and Sudholt M., "Concurrent Aspects," (GPCE 2006).

[12] Kiczales, G., et al., "Aspect-oriented programming," (ECOOP), 1997, 220--242.

[13] Bergmans L., Tekinerdogan B., Glandrup M., Aksit M., "Composing Software from Multiple Concerns: Composability and Composition Anomalies," ICSE'2000.

[14] AspectWorkz2, http://aspectwerkz.codehaus.org/, last updated on August 14, 2013.

[15] ApectJ, http://www.eclipse.org/AspectJ/, last updated on August 14, 2013.

[16] JBoss AOP, http://www.jboss.org/jbossaop, last updated on August 14, 2013.

[17] Spring AOP, org.springframework, last updated on August 14, 2013.

[18] Clifton C., Gary T., "Obliviousness, Modular Reasoning, and the Behavior Subtyping Analogy," SPLAT 2003.

[19] Shigeru C. "Load-Time Structural Reflection in *Java*," (ECOOP '00).

[20] Tennent R., "The Denotational Semantics of Programming Languages," Communications of ACM 1976.

[21] Farchi E., Nir Y., and Ur S., "Concurrent bug patterns and how to test them," Parallel and Distributed Processing Symposium 2003.

[22] Douence R., Motelet O., and Sudholt M., "A formal definition of crosscut," MISC 2001.

[23] Block Diagram, wikipedia.org/wiki/Block_diagram, last updated on February 09, 2013.

[24] Shaw M., Garlan D., "Software Architecture: Perspective on an Emerging Descipline", Publication Date: April 12, 1996, ISBN-10: 0131829572.

[25] Lopes, C. "D: A Language Framework for Distributed Programming". PhD Thesis, Northeastern University, 1997.

[26] Christakis M., and Sagonas K., "Detection of Asynchronous Message Passing using Static Analysis", PADL'11.

[27] Tulio M., et al.. "An aspect-oriented communication middleware system", (OTM'05)

[28] Dollimore J. et al.. "Distributed Systems: Concepts and Design," (4th Edition); ISBN-10: 0132143011

[29] Gamma E., Helm R., Johnson R., and Vlissides J., "Design Patterns: Elements of Reusable Object-Oriented Software," Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA. 1995.

Systematic Modeling of Workflows in Trace-Based Software Debugging and Optimization

Salman Rafiq and Adriaan Schmidt
Fraunhofer Institute for Embedded Systems and Communication Technologies ESK
Munich, Germany
Email: {salman.rafiq, adriaan.schmidt}@esk.fraunhofer.de

***Abstract*—Tracing is a tool frequently used in the debugging and optimization of software. While there exist different tracing solutions, each of them comes as a tightly coupled trace collection, analysis and visualization bundle, and thus, it can only be used to answer a narrow range of questions. Due to this limitation and the complex nature of software workflow in the embedded domain, we believe that tracing and the analysis of traces have to be flexible and extensible. In this paper, we propose a methodology of trace processing. We introduce a generic model of describing traces and operations that are performed on them, irrespective of the tracing solutions being used. Also, with the help of our model, one can describe new processes and workflows that involve trace data from a combination of sources. To present the use of our methodology, we systematically model four use cases that solve complex debugging and analysis tasks. At the end, we show how one of these use cases fits into a modular framework using a prototype implementation.**

Keywords*—*Tracing; trace-processing; workflow modeling; debugging; multicore.

I. Introduction

With the ever-increasing complexity of hardware and software systems, the task of programming and maintaining software has become more and more challenging. This is especially true when considering parallel systems, i.e., multicores and Systems-on-Chip (SoCs). To debug and optimize these systems, classical debugging tools and methods are often insufficient.

Tracing, that is the recording of data on the dynamic behavior of a software system, has been introduced with great success in some problem domains, e.g., performance optimization in High-Performance Computing (HPC) or debugging of embedded and real-time systems. So far, however, existing trace solutions cover only few specific use cases at a time. The technologies of trace collection and visualization are customized to these cases. In HPC, tools are specialized to deal with highly parallel programs, typically using the message passing model of programming. Trace analysis and graphical visualization is tailored to the task of performance optimization of such applications. In the area of embedded computing, we will find tools that can record a system's execution with hardware assistance, at a cycle-accurate level, without changing the timing behavior of the target. This data is then used for debugging and analysis purposes. Due to its non-intrusive behavior, this method is suitable for debugging timing related issues in real-time systems.

Existing trace solutions, such as [1, 2, 3], collect a vast amount of data, which is then processed and presented to the developer. However, these products mostly come as integrated solutions, tightly coupling trace collection, analysis and visualization.

With the afore-mentioned increased complexity in systems, we believe that tracing and the analysis of traces need to be flexible and easy to handle and extend. For instance, tools have to be extensible to fit complex debugging and optimization tasks. Keeping these properties in mind, we introduce a methodology to describe traces and operations on them. This methodology can be used to model different elements of a trace-based analysis and debugging workflow. Moreover, it provide ways to model complex processes that use the trace data from different tools or sources. We also present how different debugging and analysis use cases can be efficiently modeled using our methodology. By modeling a use case, dependencies between different trace data involved, and the interfaces between different tools become obvious, which helps during concrete implementation. At the end, we show how one of the modeled workflows can be mapped on to a modular framework, as part of our prototype implementation.

Previously, there has been some work on languages describing event traces. Auguston [4] suggested FORMAN language, which is used to describe computations over event traces. It uses an event grammar to define intended program behavior during debugging or testing of programs. Boroday et al. [5] presents a generalized formal framework to model event traces in a distributed system. Another language called Tiddle is proposed by Sadowski and Yi [6] to test dynamic analyses by generating concurrent benchmarks. However, these languages do not provide a way to define complex analysis workflows, that involve trace data from different sources. Also, these modeling techniques cover specific use cases, e.g., communication, or concurrency bugs, rather than being generic and scalable to other use cases as well.

Visualization tool by McGavin et al. [7] describes a methodology to explore large sets of execution traces. It gives control to the user to filter events and get on-demand information related to a particular object, after loading all the trace data into the tool. Whereas with our modular framework consisting of transformation modules, pre-filtering of trace data can be done before loading it into an output module, i.e., visualization or analysis tool. Hamou-Lhadj and Lethbridge [8] discuss different analyses and visualization tools for Object-Oriented systems, and the possibility to combine features from each tool into a common framework as their future work. The discussion centers on trace exploration and compression techniques to reduce the volume of generated traces.

The remainder of this paper is structured as follows. In Section II, we present a brief introduction to tracing and an overview of the current tracing, analysis and visualization technologies. In Section III, we describe our model and methodologies for trace data manipulation and workflow description. Section IV presents use cases which utilize our model to describe complex trace-analysis workflows. Section V shows the mapping of one of the modeled use case to a modular framework, before Section VI concludes this paper.

II. Survey of Tracing Technologies

Tracing can be seen as directly derived from one of the oldest methods in debugging: the use of print statements to output program state at runtime. The method has, however, evolved, and the use of tracing techniques yields a much more systematic approach to debugging than the simple insertion of print statements. Especially in complex parallel systems, the messages produced by manually printing program state will be hard to read and interpret. This is where tracing solutions offer graphical visualization that helps the developer to comprehend the recorded trace data.

In this section, we present a brief overview of trace-collection methods. We show the basic sources of trace events that are used as inputs to our trace-processing workflows.

A. *Instrumentation-Based Tracing*

One important method of tracing is instrumentation: additional code is added to the software at points of interest, which causes the target itself to generate event traces. These are then stored or transmitted for later analysis. There are different ways of instrumenting the target software. The most basic is manual insertion of instrumentation code by the developer. Alternatively, the insertion can be automated, often with the help of the compiler. Another class of tools performs dynamic instrumentation of the target application, that is, they change the code at run time.

The recording of event traces with the help of instrumentation can be performed in different components of a system. Aside from the application itself, instrumentation placed within the operating system can produce valuable information on system execution, capturing the interaction of several applications, together with global resources like device drivers.

Instrumentation-based trace techniques are used in different domains, and several specialized solutions are available: in the area of HPC, tools like [1, 2, 3] together with their respective visualization front-ends, are popular to optimize the performance of highly parallel applications. Instrumentation of the operating system kernel is used in debugging, but also online monitoring of systems. Solutions are available for numerous platforms, including Linux [9], Windows [10], BSD or QNX [11]. Also for embedded systems, specialized solutions are available, e.g., for low-overhead collection of events for timing analysis [12].

The advantage of using instrumentation to collect trace events is that it allows good control on the type and number of events to be collected. In this way, the volume of the generated trace can be limited to the events actually needed. Manual instrumentation of the application is easy to use, and can access application-specific data. Common examples would be the indication of program states, or the value of internal program variables.

However, instrumentation does influence the run-time behavior of the target application. So, in debugging timing-related issues, the target may exhibit a changed behavior due to the instrumentation, and the results may be useless. Also in systems that are already operating at the limit of CPU utilization may not be suitable for instrumentation, as the additional overhead may render the system dysfunctional.

B. *Hardware-Based Tracing*

Modern processors implement hardware interfaces that generate execution traces. Here, event traces are generated by the hardware and are typically on a low level, i.e., the execution of single machine instructions. The generated data is transferred off-chip via a high-speed serial interface, e.g., Serial Wire Debug (SWD) [13]. To receive the trace data, usually another hardware device (hardware debugger) is needed that decodes the event stream and transfers the data to a debugger application on the host computer. Examples of such solutions are [14, 15, 16].

A hardware trace contains detailed data on the execution of all software. In contrast, an instrumentation-based trace captures events only in the parts of the system that are instrumented, so it captures events only the developer expected. To also see unexpected events like hardware interrupts or unexpected memory accesses due to corrupted pointers, a hardware trace is much more useful. Another great advantage of hardware-based tracing is that it does not influence the timing behavior of the target system. This means that it is also suitable to debug timing issues in real-time systems.

The transfer of a target-system trace to a host computer can require extremely large bandwidths. If we take the ARM platform as an example, this can lead to a required bandwidth of 1 Gbit/s per core for a simple instruction trace and up to 16 Gbit/s per core for a complete data trace [17]. When considering multicores and SoCs with many on-chip trace sources, one quickly sees limitations in the amount of trace data that can be transported from the target to the development host. Obtaining a complete execution trace of a multicore processor is at the least challenging, and often impossible due to bandwidth limitations.

Realizing this, hardware vendors have started the integration of more-flexible tracing logic into their chips. Examples are ARM CoreSight^{TM} [18], or Infineon's MCDS [19]. With its help, it is possible to program flexible triggers and filters, and thus reduce the volume of the trace data and the bandwidth needed to transfer it.

While typically hardware-based trace solutions capture data from the processor cores, and thus data specific to the software being executed, other devices can be traced as well. Examples include traces of on-chip buses and interconnects [20] or peripheral devices [21].

C. *Other Data on Dynamic System Behavior*

Apart from traces collected using dedicated software or hardware solutions, there exist many other useful sources of

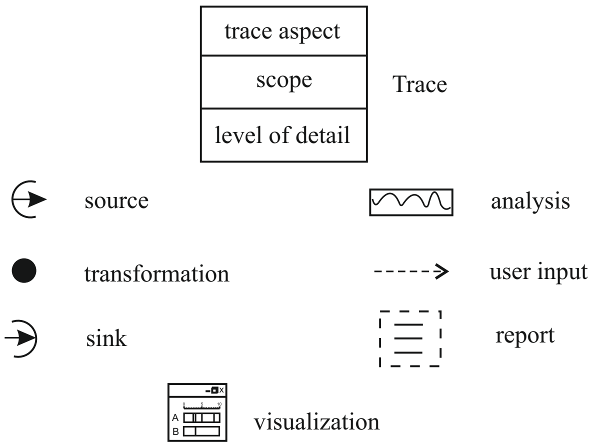


Figure 1: Elements used in the graphical representation of trace processing workflows.

data on the runtime behavior of a system. Some of them can be useful in an analysis to provide a context in which the software was executed. This is especially valuable when dealing with embedded systems that have close interaction with their surroundings.

Examples are traces of communication links, which can be captured externally using a network monitor or a bus analyzer, or information from external sensors and actuators. Also other existing sources of data present in software can be exploited in trace analysis, e.g., existing log files, providing a high-level view of the system's activities.

III. Methodology of Trace Processing

In this section, we introduce a model to describe traces and operations on them. Our objective is to provide a basis on which to argue about transformations, analyses, and the graphical visualization of trace data. Our approach can be used to describe workflows of trace processing in an abstract way, independent of a particular trace-collection technology. With the help of our methodology, complex applications in the area of trace analysis can be systematically described.

Throughout this paper, we use a simple graphical representation to visualize applications of our model. An overview of elements is shown in Fig. 1. They are described in detail in this section.

A. Trace Features

Traces are the central artifact of our model representation. Our model annotates a trace with three features: the Aspect captured by the trace, the Scope of the trace data, and its Level of Detail. We do not model the details of the trace data itself, or impose any restrictions on trace formats and representations. For the purpose of our model description, a basic definition of traces is sufficient. Thus, we define a "trace" to be an ordered sequence of events. The order may be established by associating a timestamp with each event. Additionally, trace events may contain arbitrary data; again, there is no restriction imposed by our methodology.

We do not provide a formal language for the description of trace features in our model, but rather focus on the high-level representation of traces using plain English. It is up to the developer, how detailed this description should be. In our graphical representation, a trace is depicted as a rectangular box with three fields, one for Aspect, Scope and Level of Detail.

1) Trace Aspect: The trace Aspect describes which properties of the target system are captured in the trace data. It is the most important of our three properties, as it describes the nature of the respective trace. The Aspect of a trace determines which questions can be answered by interpretation of the trace data, and it is often closely linked with the method of trace collection.

Examples of trace Aspects can be "program execution flow, i.e., which instructions or functions were executed at which time", "program state, i.e., the values of variables over time", "packets seen on a network link", "transactions on an internal bus", "performance metrics", and "inter-process communication".

2) Trace Scope: The Scope of a trace describes which parts or which components of a system are covered by the trace. Components of a system may be hardware devices, such as CPU cores or communication interfaces, or software entities, such as applications, threads, or objects.

In trace analysis, it is important that the trace data captures the right scope. For efficient operation, the trace should not contain more or less information than is needed to answer the developer's questions. If the scope is too broad, it may be difficult to grasp the essential information, and if it is too narrow, interactions between several components may be lost from the trace.

As the selection of a Scope for a trace defines which subset of the set of all available events is contained within the trace, it directly influences the volume of the resulting trace. Thus, the trace Scope determines the bandwidth required for transmission of the trace, or the capacity needed for its storage. The possible values of the Scope feature naturally depend on the Aspect captured by the trace. When considering a trace of the program execution flow, the Scope may be, for example, "instructions executed by application A", "instructions executed in interrupt service routines", or "instructions executed on Core n". In contrast, a trace of network packets or protocols may have scopes like "TCP, HTTP, or telnet session", "connection in a client-server scenario" or "communication link".

3) Level of Detail: The Level of Detail (LoD) gives information on the resolution or precision of the trace. It can be described as a set of information captured to fulfill the requirements in achieving a certain trace Aspect.

Similar to the Scope feature, the LoD in a trace depends upon the trace Aspect. Also it influences the trace volume as in the case of a Scope. Carefully choosing the LoD can simplify the trace collection, transmission, and storage in cases where the lower LoD is tolerated by the analysis in question.

For example, a trace Aspect "execution flow" may have LoD like "complete set of instructions", "only branches", or "only function entries and exits". On the other hand trace Aspect "data values" can have LoD like "modifying a certain memory location", "reader or writer process/thread", or "reads/writes within a specific region of the code".

B. Basic Trace Transformations

A trace transformation is a step of processing, which takes one or more traces as input, and based on them generates one or more traces as output. Within our model, a transformation always affects at least one of the three trace properties.

The effects of some operations are easy to capture and understand. For example, filtering operations will usually either narrow the Scope of a trace, or lower its Level of Detail. Examples would be the processing of a hardware trace containing all executed instructions through a filter, which reduces the data to include only the executed calls and returns, thus yielding a lower LoD (function entry/exit instead of instruction accurate). Also, a reduction in Scope is common, and can be seen as a filter operation performed on the trace. An example is the filtering of a trace of scheduling events of the operating system, to extract only those events related to a certain application. These two operations are easily implemented as filters, extracting some events from a trace and discarding the others.

Other transformations are more complex. Among them are: broadening of a Scope; which combines traces with distinctive features. This combination of different traces with distinctive features has the potential of substantially increasing the value of tracing. In systems with complex interactions of different components, it is often necessary to combine data from many different sources to track down the origin of a software failure. If this integration of data sources can be performed systematically, as opposed to the developer switching back and forth between tools, debugging and optimization can be greatly simplified.

Changing the Aspect of a trace is another complex transformation, which can require some more-elaborate analysis and processing. A complete trace of program execution could be processed, such that instead of the execution flow of the application it reflects the value of certain variables over time.

In the processing of hardware traces, there is a commonly used transformation that increases the level of detail. The trace unit of modern CPUs compresses the trace data to reduce bandwidth. Usually, only the branch instructions are captured, knowing that from this information the instructions executed between branches can be reconstructed. This step is implicitly performed by the host-side tracing tools.

C. Trace Sources and Sinks

To model trace-processing workflows, it is not only necessary to describe trace transformations, but also to specify where traces come from (their sources) and what eventually happens to them (their sinks). Trace sources are means of trace collection, of which we already described several in Section II. We distinguish two kinds of sinks: visualization and analysis.

The trace sources are the point where traces enter our modeled workflows. The trace-collection method determines the features of the trace at this point. Multiple trace sources can be employed in one workflow, e.g., combining traces that capture different Aspects or Scopes of the same system.

It is important to mention that workflows described by our model can have many processing steps, not all of which will be actually implemented in software. It is possible for some of the steps to be implicitly performed in hardware, e.g., on-chip filtering of trace events. In these cases it is up to the developer, whether this implicit processing step is modeled, or whether the model uses the already filtered data as a trace source.

A visualization presents the trace data to the user, showing the trace events on a time line. On the other hand, an analysis takes a trace as input and generates results in arbitrary form. Their semantics differ from the transformations described earlier, in that their output is not a new trace. Thus, the analysis result, which might take the form of a textual or graphical report, lies outside the scope of our model. Examples of analyses are: timing properties like average execution times of functions or distributions of response times, analysis of lock contention, or analysis of data accesses to detect race conditions.

Both visualization and analysis can present data to the developer in an interactive way. It may be possible to "browse" the trace graphically, scroll, zoom, etc. User interaction can influence the processing inside the modeled workflow. By selecting elements from the analysis or visualization, the user could change the parameters of certain transformations, causing a re-calculation of the results. Such an influence on the processing of traces is represented by a dashed arrow in our graphical representation.

IV. Modeling of Trace-Analysis Workflows

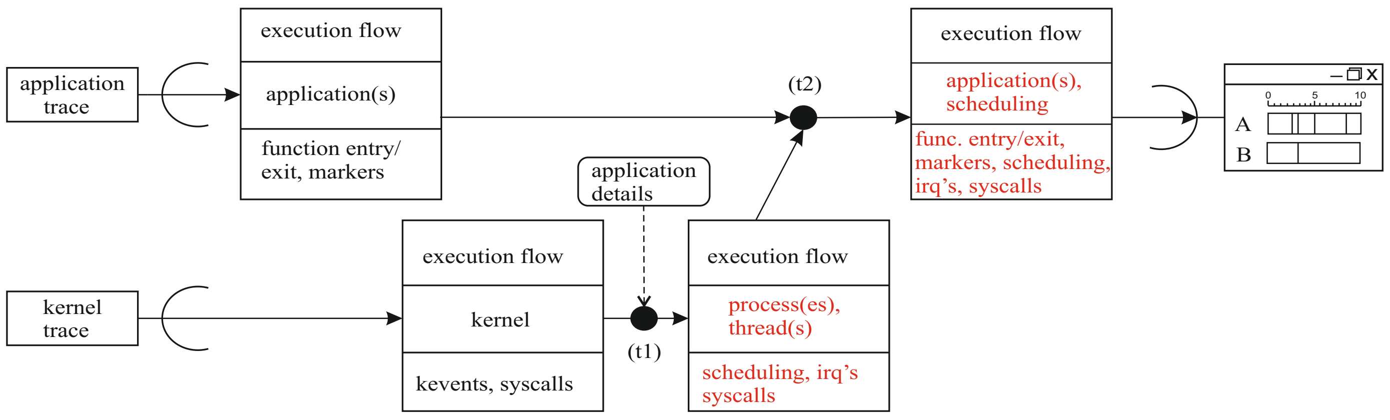
In this section we present four examples, how our methodology can be applied to model certain analysis workflows. In addition to describing complex workflows, our methodology can be utilized to constitute a flexible modular framework. The sources providing trace data as an input, translate into input modules. Also the user interaction becomes part of an input module. The sinks utilizing trace data for visualization and analysis are mapped as output modules. Finally, trace transformations that involve processing of traces are implemented as transformation modules.

In the following section, we present the prototype framework implementation using one of the described examples. For the rest of the examples, our model documents trace-processing workflows in an abstract way, and does not concern itself with concrete implementations.

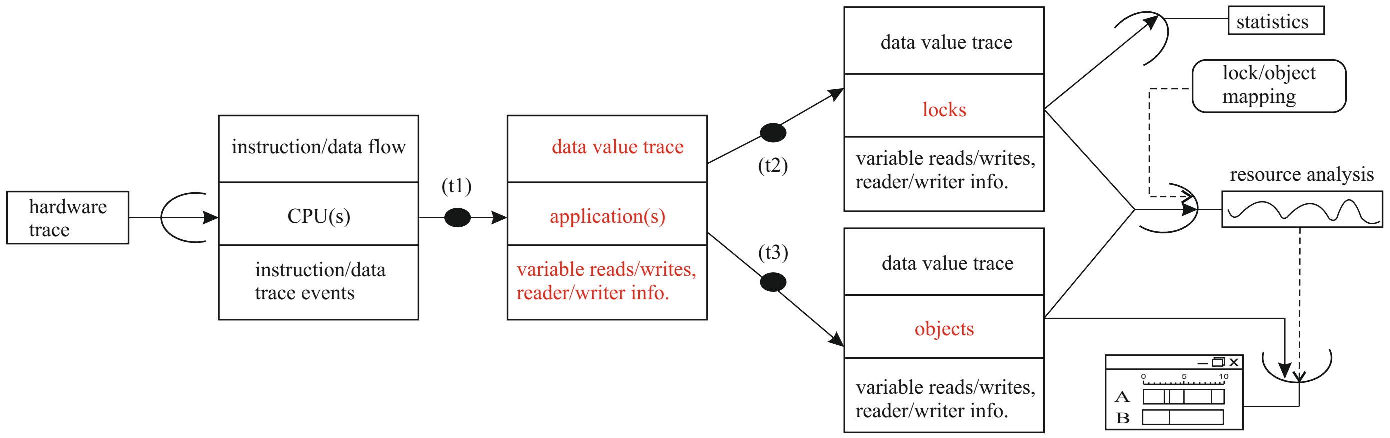
A. Combination of Application and Kernel Trace

Manually instrumenting a target application is an efficient way of collecting trace data. The developer can select instrumentation points, and can thus easily create a trace containing application state, phases in program execution, and values of important variables. However, one weakness of this approach is that it can capture only the single application that is instrumented. Especially in embedded systems, this is often insufficient, and instead the developer needs a view of the complete system, showing interactions between different applications, and between applications and the operating system.

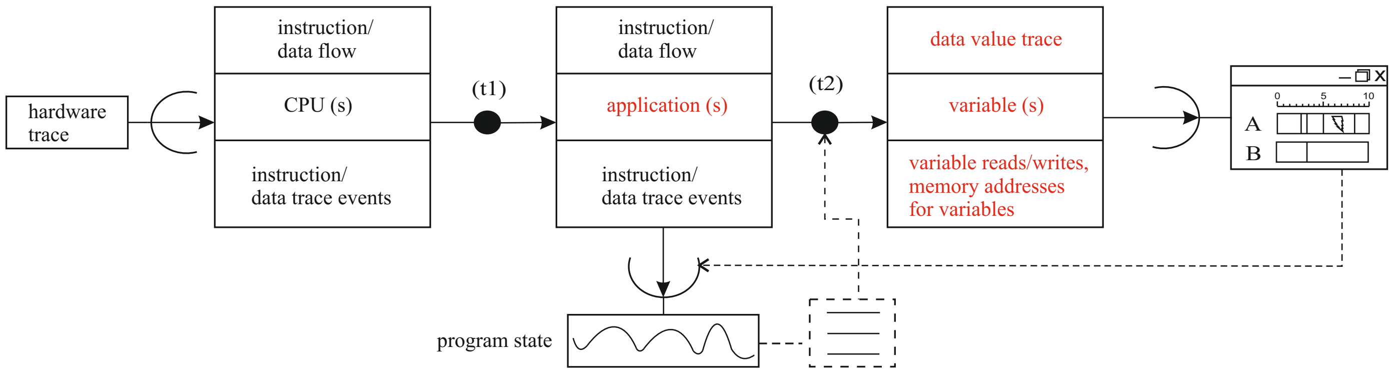
The first use case of our methodology addresses this issue, by processing separately collected traces from the user application and the operating system kernel, and integrating



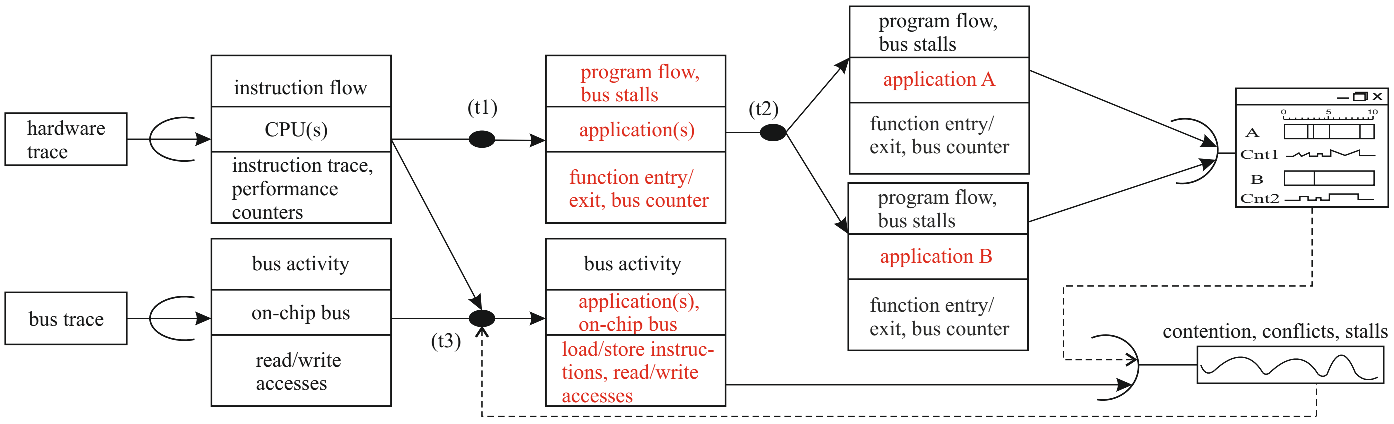
(a) Use case described in Section IV-A. Combination of application and kernel traces.



(b) Use case described in Section IV-B. Modeling of trace features for data-centric analysis.



(c) Use case described in Section IV-C. Modeling analysis workflow of memory related issues.


hardware trace
instruction flow
CPU(s)
instruction trace, performance counters
(t1)
program flow, bus stalls
application(s)
function entry/ exit, bus counter
(t2)
program flow, bus stalls
application A
function entry/ exit, bus counter
program flow, bus stalls
application B
function entry/ exit, bus counter
A
Cnt1
B
Cnt2
bus trace
bus activity
on-chip bus
read/write accesses
(t3)
bus activity
application(s), on-chip bus
load/store instructions, read/write accesses
contention, conflicts, stalls

(d) Use case described in Section IV-D. Integration of on-chip bus trace for enhanced analysis.

Figure 2: Example use cases

them into one combined view. The workflow is shown in Fig. 2a.

The essential transformation (t2) in this workflow performs a “broadening” of trace Scope by combining the execution flow of the application with the relevant scheduling events from the kernel trace. This way, the application’s timing can be displayed in the context of the complete system, and in case of timing anomalies a picture of the system state with the current constellation of running applications and occurrences of interrupts is available.

While the application trace is directly used by transformation (t2), the kernel trace first undergoes some filtering: from a kernel trace containing a multitude of different events, we extract only events related to scheduling, system calls, and interrupts that affect our target application as shown by transformation (t1). This transformation also uses auxiliary application information that helps in mapping of target application to kernel trace.

Note that the trace Aspect remains unchanged by all transformations. We always consider the program execution flow. Also, the Level of Detail is largely kept unchanged. All processing and analysis is done on a relatively high level of abstraction, considering program phases and single scheduling events, but not further details on instructions executed.

B. Extraction of Data-Centric Trace Information

In object oriented programs, it can become useful to keep track of an object’s lifetime. It starts from the object creation, different processes/threads accessing it, interactions with other objects, and finally deletion of the object. Moreover, the inclusion of locks to protected shared objects makes interactions more complex and difficult to analyze. With the help of such an analysis, one can find anomalies in software like data races (simultaneous access to an object with at least one of them being a write operation), locking violations, contention for guarded shared objects, and memory leaks (objects created but never deleted).

Hardware tracing allows the developer to generate cycle-accurate event traces. It can include a complete set of instructions, and reads and writes to memory addresses. This use case of our methodology uses hardware tracing as a source, and reduces the LoD to specific objects and locks with the help of the developer. The workflow is shown in Fig. 2b.

The transformation (t1) filters the Scope of the hardware trace from the CPU to the target application. Furthermore, (t1) also performs an important transformation of Aspect of the trace to “data value trace”. This is achieved by extracting the selected LoD from instruction and data-trace events, e.g., memory reads and writes, and reader/writer process or thread information.

Later, the transformations (t2) and (t3) are performed, generating new traces with similar Aspects as before, i.e., data value trace. These transformations reduce the LoD to specific shared objects and locks inside the application, rather than a complete set of variables. From this point on, the transformed trace can be used as an input to the trace-analysis tool. The user may also provide auxiliary information regarding the relation between a lock and an object to analyze them over time. This mapping helps to find any problems with regard to simultaneous accesses from different writers/readers and the locks guarding them.

The trace data with object as a Scope can then be used as an input to the visualization tool for a graphical representation of the object’s lifetime. The dotted arrow (user input) from the analysis tool to the visualization sink shows that the user can also influence the graphical time line of an object of interest.

C. Backtracing of Memory Issues

Root causes of memory-related faults become more difficult to find, especially when software is running on a multicore system with shared memory resources. Consider a case where an incorrect value is being assigned to a memory location, causing the program to become dysfunctional. It can be of great help if a backward chain of calculations can be analyzed to know the source code location causing that faulty write. To achieve this, it requires the program state to be maintained by the tool for that point in time. This use case addresses workflow-modeling of a similar scenario.

For example, the event traces of specific variables are visualized in a tool showing data values being assigned to them over a time period. The tool allows the user to inquire about the incorrect value on demand. This interaction from the user triggers the backward analysis by inquiring the program state being maintained by the analysis tool. Finally, an interactive report provides the instruction writing the wrong value to the variable. Fig. 2c explains the steps involved to model such an analysis with the help of a hardware trace source.

It is important to mention that transformation (t1) may be implicitly performed by configuring a hardware trace-recording tool to record events related to a single application only. In this case the model will directly contain trace data with “application” as the Scope. Information like memory addresses, data reads/writes, and program-counter values acquired from instruction and data trace are used by the analysis tool to maintain the program state.

The next transformation (t2) changes the Aspect of the trace to “data-value trace” and filters the Scope to a particular variable in an application using variable reads/writes, address of variable in memory etc. as LoD. Finally, the visualization tool uses this trace to show variable data values over time.

The dotted line from the visualization tool represents the user interaction. The analysis report from the program state further represents the subpart of the analysis tool that is interactive and allows the user to influence the transformation (t2) for a refined visualization.

D. Incorporation of Data from on-chip Bus Trace

Since embedded systems are composed of a set of different components, traces from buses, peripherals, and controllers can add value for a comprehensive analysis. Specially by combining an event trace containing all bus transactions with an instruction trace can provide information related to program read and write accesses, thus also broadening the Scope.

The only problem with such a combination is that it requires the target to export all the required information to clearly map load/store instructions to read and write accesses. This can become difficult with limited bandwidth of trace ports.

This use case presents a similar scenario where combining hardware and bus traces can help the user to investigate unusual bus activities. The user can interactively examine a region in the visualization tool that shows a high number of average bus stalls. This inquiry leads to the analysis tool which, with the help of the combined trace, maintains reports related to program read and write accesses, contention among different bus transactions or any conflicts. For instance, the user may find the application or source-code location that is causing the extra stalls by holding the bus, after looking into this analysis report. Moreover, it can be used to find if the CPU is waiting for a response from a peripheral device that also shares access from another application. Fig. 2d provides the model of this workflow.

Hardware trace in this case contains an extra LoD: Performance counters, which are hardware registers commonly available in most CPUs for measuring performance metrics like cache misses, instruction count, cycle count, bus stalls and so on. The transformation (t1) changes the Aspect of the trace from "instruction flow" to "program flow" by extracting the data events related to executed calls and returns, thus lowering the level of detail to function entry/exit.

Transformation (t2) further filters down the trace to selective target applications and their respective LoD, i.e., function entry/exit and counter values for bus stalls. The visualization tool then can be used to graphically present function entries/exits, along with the average bus-stalls histogram for the duration of the function execution. A similar workflow can also be modeled using instrumentation-based tracing as a source only.

In case of higher bus-stall cycles causing longer executions for a function, the user can select that region from the histogram for a detailed analysis. This is shown by the dotted line from the visualization tool which influences the information used as input to the analysis tool.

The transformation (t3) takes data like load and store instructions from the instruction trace and maps this information with bus transactions to get the trace with application-specific reads and writes. The analysis tool uses this trace data to provide a more-detailed analysis related to any conflicts, contention, and stalls over the system bus. Moreover, the user can influence the transformation (t3) for a refined analysis represented by the dotted arrow from the analysis tool.

V. Case Study Implementation: Integrated Application and Kernel Tracing

The objective of this section is to present how a modeled workflow can be translated into a modular framework application. In order to support this mapping, we implemented the first example described in the previous section as a prototype. This example integrates different trace sources, not only to broaden the scope but also to provide a comprehensive visualization for the developer. It shows how this improved visualization can help to find sporadic errors in an application.

A. *Prototype implementation*

As a proof of concept, we began with the implementation of combined application and kernel tracing. We chose VampirTrace [1] for the application and LTTng [9] for kernel tracing respectively. Event traces from VampirTrace were stored in the Open Trace Fromat (OTF) [22], and in the Common Trace Format (CTF) [23] from LTTng.

We manually instrumented two applications using VampirTrace for entry and exit of different phases (functions). One of these applications was periodic with timing constraints (soft-RT) task, while the other was non-periodic and without timing requirements (GPtask). For the soft-RT task, a marker API was also added to get the events for the cases in which the application may miss any deadline. Both of these applications were scheduled on separate cores (core affinity). At the same time with the help of LTTng, kernel events (system calls, interrupts and scheduling) were recorded.

The OTF streams generated by the applications were then fed as an input to the VAMPIR [24] visualization tool to visualize the program execution flow. Fig. 3 shows a time slice of soft-RT and GP application events.

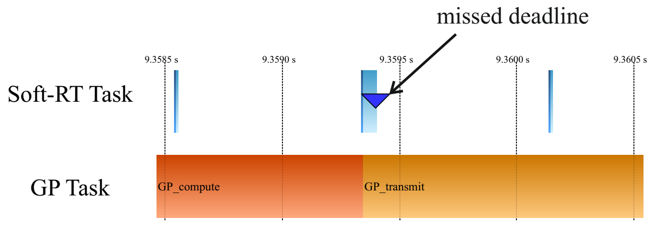


Figure 3: VAMPIR screen-shot: Visualization of application traces for real-time and general-purpose tasks.

The two different colors in soft-RT task's timeline represent different functions being executed, whereas the periodic nature of the task can be seen from invocations of these two functions at distant time intervals. Also, there is a triangle on top of one of the invocations, indicating a missed deadline. The lower time line indicates two different phases of the GP task. With this view in the visualization tool, it is difficult to speculate about any reason for the missed deadline.

For our prototype framework implementation, we used a modular approach in mapping sources, transformations, and sinks. Since the trace data in this use case comes from different sources with different trace formats, we chose OTF as an internal format for the framework. For this reason, the kernel trace which is stored in the CTF format is converted into the OTF format.

The input module provides CTF traces depicting the execution flow of both applications to the transformation module. It then performs the necessary processing to reduce the Scope of the trace to application specific kernel events. This module also correlates the kernel and application traces with the help of timing information, and converts the kernel trace from CTF to OTF. After the conversion, another independent transformation module merges the application trace (input module) with the transformed kernel trace, in order to prepare it for the output module (VAMPIR tool).

Fig. 4 shows a screen shot from the tool with integrated kernel tracing using the transformed trace information, as shown previously in Fig. 2a. The additional two time lines indicated

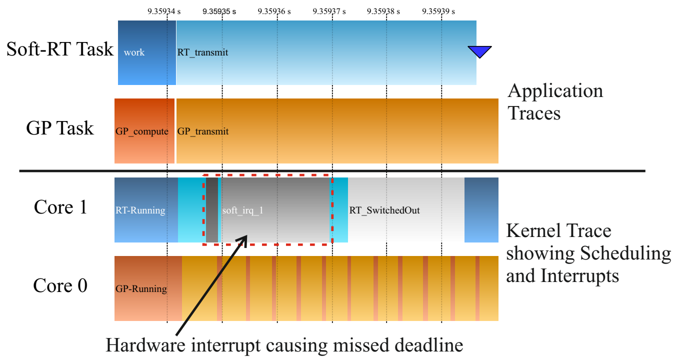


Figure 4: VAMPIR screen-shot: Combined application and kernel scope with zoom-in view for missed deadline.

by Core 1 and Core 0 represent the actual mapping of tasks onto the hardware. In this specific case, the soft-RT task was scheduled on core 1 and the GP task on core 0. Moreover, it can be seen that now the timeline shows extra events like hardware/software interrupts and scheduling events, e.g., thread switch in/out during the same execution period. In other words, the correct notion of function execution time is being depicted using kernel trace data.

Finally, by looking into the integrated view, the user can now identify the actual reason behind the soft-RT task missing its deadline, which in this case is caused by a hardware interrupt being serviced by Core 1.

VI. Conclusion and Future Work

The methodology introduced in this paper can be used to systematically describe complex analysis workflows. The possibility of constructing methods for complex workflows, which can utilize trace data from multiple sources, can be used to develop flexible tools for trace-based debugging and optimization. Our goal is to encourage the use of existing sources of trace data.

With the help of provided modeling notations, one can document the new processes and workflows in an abstract way. By modeling a workflow, the dependencies between different trace data involved and the interfaces between tools become transparent for the implementation. Our methodology can be used not only for modeling purposes, but also for providing a basis for mapping the modeled workflow to a flexible and extensible framework. We have shown this by translating one of the use cases to a modular framework in our prototype implementation.

As part of our future work, we intend to extend the framework with an internal format other than the OTF. Also, the trace features and transformations that are modeled using simple graphical notations will be described formally with the help of a machine-readable language. Furthermore, the details about the trace data (events and their semantics) will be represented using our model. These additions will help in automation of the trace processing. Finally, the modeled workflows will support automatic generation of "glue code" for a framework implementation.

References

[1] A. Knüpfer, H. Brunst, J. Doleschal, M. Jurenz, M. Lieber, H. Mickler, M. S. Müller, and W. E. Nagel, "The vampir performance analysis tool-set," in *Tools for High Performance Computing*. Springer, 2008, pp. 139–155.

[2] V. Pillet, J. Labarta, T. Cortes, and S. Girona, "Paraver: A tool to visualize and analyze parallel code," *WoTUG-18*, pp. 17–31, 1995.

[3] F. Wolf, B. J. Wylie, E. Abrahám, D. Becker, W. Frings, K. Fürlinger, M. Geimer, M.-A. Hermanns, B. Mohr, S. Moore *et al.*, "Usage of the scalasca toolset for scalable performance analysis of large-scale parallel applications," in *Tools for High Performance Computing*. Springer, 2008, pp. 157–167.

[4] M. Auguston, "Building program behavior models," *Engineering Automation for Reliable Software*, p. 35, 2000.

[5] S. Boroday, H. Hallal, A. Petrenko, and A. Ulrich, "Formal modeling of communication traces." in *ISTA*. Citeseer, 2003, pp. 97–108.

[6] C. Sadowski and J. Yi, "Tiddle: a trace description language for generating concurrent benchmarks to test dynamic analyses," in *Proceedings of the Seventh International Workshop on Dynamic Analysis*. ACM, 2009, pp. 15–21.

[7] M. McGavin, T. Wright, and S. Marshall, "Visualisations of execution traces (vet): an interactive plugin-based visualisation tool," in *Proceedings of the 7th Australasian User interface conference-Volume 50*. Australian Computer Society, Inc., 2006, pp. 153–160.

[8] A. Hamou-Lhadj and T. C. Lethbridge, "A survey of trace exploration tools and techniques," in *Proceedings of the 2004 conference of the Centre for Advanced Studies on Collaborative research*. IBM Press, 2004, pp. 42–55.

[9] M. Desnoyers and M. R. Dagenais, "The lttng tracer: A low impact performance and behavior monitor for gnu/linux," in *OLS (Ottawa Linux Symposium)*. Citeseer, 2006, pp. 209–224.

[10] I. Park and R. Buch, "Event tracing- improve debugging and performance tuning with etw," *MSDN magazine*, p. 81, 2007.

[11] T. Beauchamp and D. Weston, "Dtrace: The reverse engineer's unexpected swiss army knife," *Blackhat Europe*, 2008.

[12] N. Merriam, P. Gliwa, and I. Broster, "Measurement and tracing methods for timing analysis," *International Journal on Software Tools for Technology Transfer*, vol. 15, no. 1, pp. 9–28, 2013.

[13] M. Williams, "Low pin-count debug interfaces for multi-device systems," 2009.

[14] (2013, August) LAUTERBACH Development Tools. [Online]. Available: http://www.lauterbach.com/

[15] (2013, August) PLS Development Tools. [Online]. Available: http://www.pls-mc.com/

[16] (2013, August) iSYSTEM. [Online]. Available: http://www.isystem.com/

[17] W. Orme, "Debug and trace for multicore socs," ARM Limited, White Paper, 2008.

[18] "Coresight^{TM} components technical reference manual," ARM Limited, Tech. Rep., 2009.

[19] N. Stollon, "Infineon multicore debug solution," in *On-Chip Instrumentation*. Springer, 2011, pp. 219–230.

[20] "Amba ahb trace macrocell (htm) technical reference manual," ARM Ltd., San Jose, CA, Tech. Rep. ARM DDI 0328E.

[21] "Usb event tracign for windows," Microsoft, White Paper, March 2010.

[22] A. Knüpfer, R. Brendel, H. Brunst, H. Mix, and W. E. Nagel, "Introducing the open trace format (otf)," in *Computational Science–ICCS 2006*. Springer, 2006, pp. 526–533.

[23] (2013, August) Common trace format (ctf). EfficiOS. [Online]. Available: http://www.efficios.com/ctf

[24] W. E. Nagel, A. Arnold, M. Weber, H.-C. Hoppe, and K. Solchenbach, *VAMPIR: Visualization and analysis of MPI resources*. Citeseer, 1996.

A Pattern-based Approach towards Expressive Specifications for Property Concepts

Geert Delanote, Jeroen Boydens and Eric Steegmans
KU Leuven, Dept. of Computer Science
Leuven, Belgium
{geert.delanote, jeroen.boydens, eric.steegmans}@cs.kuleuven.be

***Abstract*—In Object-Oriented programming, a significant effort has been made in recent years to increase the expressiveness of programming constructs for the production of code. Developers can implement more functionality in less lines, and with more compile-time guarantees. We have not seen such a similar evolution in the design and specification of code. Support for code specification remains a feature that is rarely integrated in the language itself (e.g., Eiffel), and is too often migrated to ad hoc language additions (e.g., annotations). The lack of such first-class, language-integrated support leads to (1) developers who are forced to write ad-hoc code specifications in a non-standardized manner, often ex-post and time-permitting, and (2) to situations in which other developers, who reuse that code, are tempted to read the code itself (if available) rather than the specification, in order to understand what the code actually does. In this paper, we take an evolutionary approach to language-integrated specification constructs, with the ambition to enhance the overall expressiveness of specifications in object-oriented languages. We start from existing best practices and propose improvements through specification patterns that not only enhance the expressiveness of specifications, but also aid developers in specifying their code through concrete "structures" in order to avoid ad-hoc, non-standardized specifications. Finally, we also propose language constructs that help aim to increase the level of abstraction, by shielding developers from boilerplate specification as much as possible.**

Keywords*—*Pattern; Specification; Property; Language Construct.

I. Introduction

Object oriented programming languages use classes as abstract data types [1][9]. A class is a blueprint for a collection of objects with identical characteristics and behavior. Encapsulation hides the technical details of the data fields used in the implementation to describe those characteristics. Generally, several requirements have to be enforced for those characteristics. Examples of such requirements are: the balance of a bank account must not exceed the credit limit, a single transaction must not change the balance with more than €1000 and the holder of a bank account must be adult. Programming language constructs lack expressiveness to describe those requirements in an integrated way.

In this paper, we present a pattern to implement characteristics with their requirements in Java. We identify different kinds of requirements and show how they are implemented by the pattern. The pattern is only worked out for properties in this paper. However, with some adaptations to meet the specific needs, the pattern can also be used for (bidirectional) associations. We will show how the pattern improves the quality of the code. Finally, we will also show a new language construct that can replace the pattern.

This paper is structured as follows. Section II defines the quality objectives we want to improve. Section III presents some general programming principles to improve the quality. The different kind of requirements related to the development of properties are described in Section IV. Section V shows how the different requirements are developed in the pattern and how the pattern improves the quality of the code. In the last paragraph, a Language Construct that improves the expressiveness of a programming language is presented. Section VI presents related work. We conclude in Section VII with a view on future research roadmap.

II. Objectives

Object-Oriented languages were initially built to increase the quality of software applications [6]. Software quality is a combination of several factors [1]. Using software patterns is an important way to increase the quality of software systems [2]. We believe that more expressive language concepts can help to further improve the quality of software systems. Therefore, we believe that, as a second step, patterns should be transformed as much as possible into language concepts to avoid known drawbacks from patterns like implementation overhead (boiler plate code) and reusability (the programmer is forced to implement the pattern over and over again) [5]. Software quality factors break down in external and internal factors. In this paper, we mainly focus on the internal factors: factors perceptible for programmers. In the end, only external factors count, but the internal factors make it possible to obtain them [1]. We have centered the specification and development of our pattern along the following quality factors.

O1 - Correctness. Software must perform its task as defined by the specification. The pattern defines specific methods to work out the different aspects of the implementation of a characteristic forcing the developer to think about each aspect in isolation.

O2 - Extendibility. Software must be adaptable to future changes of the specification. These changes can be in space (through adding a subclass that redefines some aspects) or in time (changes to specification in the future). The pattern provides the necessary hook methods to be able to change the specification easily. The pattern also guides the developer to specify and implement each aspect only once.

O3 - Testability. Testing the correctness of software must be as easy as possible. Different aspects of the implementation of a characteristic are worked out in separate methods. The methods are designed in such a way that they can be tested in isolation.

O4 - Understandability. A programmer must understand as easy as possible the source code of a software system. Dividing a big problem into smaller problems is a well-known strategy to make a problem easier to understand. The pattern separates the code, the developer has to write, from boilerplate code to make the code more readable.

O5 - Reusability. Software should be usable in different applications. Extendibility already mentions the provided hook methods to change the specification easily. These methods make it also easy to reuse the software in a (slightly) adapted form in another application.

O6 - Expressiveness. The ease for a developer to write software. By forcing the developer to implement the different methods, the pattern also guides the developer through the different aspects of implementing the characteristic. This way the developer can think more on *what* must be implement instead of on *how* he can accomplish it.
We raise the ambition level for each of these objectives when compared to the current state of the practice (throughout this paper, we will refer to these objectives using their codes). Our language concept, resulting from this pattern, also meets these objectives.

III. Principles and Notation

In this paper, we will follow the principles and notations introduced in the book *Object Oriented Programming with Java* [9]. The book presents three different paradigms to deal with exceptional circumstances: nominal, defensive and total programming. Nominal programming uses preconditions to prohibit method invocations under exceptional conditions. Defensive programming uses exceptions to signal that methods have been invoked under exceptional conditions. Total programming turns exceptional conditions into normal conditions. We have chosen to work out the examples in this paper in a defensive way. Transformation to the other paradigms is straightforward.

P1 - Inspector-mutator principle. An important principle is that we make a clear distinction between *inspectors* and *mutators*. Inspectors return information about the state of some objects. Mutators change the state of some objects. We try to avoid methods that combine both aspects: inspectors should not change the observable state of one or more objects and mutators should not return a result. We further distinguish between basic queries and derived queries. A basic query returns part of the state of an object. The state of an object is determined by the set of all basic queries. The result of derived queries and the effect of mutators is directly or indirectly specified in terms of basic queries. This principle improves the quality factors described in objectives O2, O3, O4, O6.

P2 - Steady versus Raw state. We distinguish between a *steady state* and a *raw state* for objects. An object in steady state satisfies all its invariants. An object in raw state is not guaranteed to satisfy all its invariants. Unless explicitly stated otherwise all objects must be in the steady state upon entry to and exit from a method. The general principle in defining methods is to assume that all objects are in the steady state. However, in some specific situations we want to use methods that involve objects that are in raw state. A typical example of such a situation is construction. While not yet in a steady state we sometimes want to use other methods during the initializing process. This principle acts as a contract between the developer and user of a method and by doing so helps to improve quality factors O1, O3, O4, O6.

P3 - Liskov Substition Principle. Changes to the definition of inherited methods must obey the Liskov Substition Principle [3]. Broadly speaking, the principle states that it must always be possible to substitute an object of a superclass by an object of its subclasses. Next to changes to the signature of inherited method, changes to the specification can be made if the superclass does not provide a deterministic specification of the result. Non-determinism plays a crucial role in our pattern. This principle supports all objectives O1-O6.

P4 - Complete business logica. All business rules should be worked out in specification and implementation. For enforcing business rules we never rely on the underlying persistence level. Integrity constraints, non-null constraints, foreign keys, etc., can be enforced by a database, but should (also) always be enforced by the application. This principle improves objectives O1, O3, O4, O5.

Notation. In Java, the contract of a class is worked out in documentation comments, which can be processed by javadoc [15]. Tags structure the different pieces of the specification in the documentation. The specification of a class is described both formally and informally. The informal specification is written in natural language, while Java boolean expressions are used to write the formal specification. Writing the specification formally improves the objectives O1, O3, O4. The following tags are used in the code snippets throughout this paper:

- @basic: denotes a basic query
- @effect: specifies the semantics of a mutator in terms of another mutator
- @invar: denotes a class invariant
- @post: specifies a postcondition of a mutator
- @raw: denotes an object in a possible raw state
- @return: specifies the result of a derived query
- @throws: specifies the exception that must be thrown when the specified assertion evaluates to true

IV. Requirements

Business rules can be generally described using three types of requirements: (1) Value Requirements, (2) State Requirements and (3) Transition Requirements.

Value Requirements. (VR) These requirements are used to specify the most basic kind of business rules in that they restrict the range of values that a characteristic, property or association, can have. Meeting its value requirements is a necessary condition for an object to be in a steady state (P2). A value requirement never takes into account other characteristics of the class at stake. For *properties* a value requirement restricts the set of values offered by its type further. A value requirement is for instance used to enforce that the credit limit of a bank account always needs to be below 0. For *associations* a value requirement restricts the multiplicity of an association. The requirement that a bank card always needs to be linked with a bank account is a value requirement (this type of requirement is also know as *existential dependency*). Considering generalization/specialization, a redefinition that restricts the kind of objects a specialization can be linked with is a also a value requirement. The requirement that current accounts and savings-accounts, specializations of bank account, have the right specialization of bank cards attached to it is enforced with a value requirement.

State Requirements. (SR) Mostly, business rules restrict possible values for a characteristic when considered in combination with values from other characteristics. State requirements are by nature *symmetric*. A state requirement involving characteristics α and β is always a state requirement for both characteristics. Meeting its state requirements is the other necessary condition for an object to be in a steady state (P2). The union of all value requirements and state requirements describe all the invariants of a class. The business rule stating that the balance of a bank account must never be below the credit limit is specified by a state requirement.

Transition Requirements. (TR) These very specific requirements specify the business rules that restrict the evolution of values of characteristics. It's perfectly possible that a (new) value for a characteristic meets all value and state requirements but is not acceptable because of the current state of the object. The business rule imposed by a bank limiting the amount of money that can be withdrawn from a bank account is transition requirement. Although 1.000 euro is a correct balance, it's not an acceptable balance after a withdraw operation when the current balance is 10.000 euro and the withdraw limit is 5.000 euro.

In the remainder of this paper, we will show how our pattern implements the value, state and transition requirements. We will prove how distinguishing between these kinds of requirements together with the pattern with its specific methods meets the targeted objectives. We will also discuss how Java (and other object-oriented programming languages) can be extended with new language concepts to capture value, state and transition requirements.

V. Properties

In this section, we build the pattern for properties, step by step. These steps already give a good indication of what an iteration of the development process can consist of. It is possible to elaborate the different requirements independent of each other (O1, O3, O4, O6). Typically a pattern contains boilerplate code, we will highlight those parts in the code listings. The code editor should generate this code (O1, O6). In Eclipse [17], custom templates can be defined to generate skeletons of methods. Due to space limitations we omit the informal specifications. Steegmans illustrates in [9] how informal specifications should be added.

The example used throughout the next paragraphs describes a class of bank accounts. Each bank account has two characteristics, namely a balance and a credit limit. Both characteristics are decimal values and the balance must never be less than the credit limit. The amount of money that can be deposited or withdrawn in a single transaction must be restricted to 1000. To explain the pattern in the context of inheritance, we introduce a class of junior bank accounts, a subclass of bank accounts. The balance and credit limit of junior bank accounts are restricted to integer values. At the level of the subclass, two new characteristics are introduced: each junior bank account has an integer value as upper limit and a blocked state (boolean). While the credit limit can no longer be less than -1.000, the upper limit must at least be 1.000 and must not exceed 10.000. The upper limit is an immutable characteristic. Of course, the balance is not allowed to exceed the upper limit.

Representation. Each observable characteristic is part of the state of an object and is revealed by a basic query. The basic query can be compared with the getter from Enterprise JavaBeans (EJB) [10], [16]. The return type of the basic query reveals the chosen type for the characteristic. The characteristic can internally be stored using one or more *instance variables* with the same or different types. The implementation of the basic query has to perform necessary transformations between stored and observable information. Like EJB, we introduce also a *setter* to change the characteristic to a given value. The basic query and this setter are the only two methods that are allowed to access the instance variables that represent the characteristic. By consequence, we limit the optional transformations between internal representation and observed value of a characteristic to these methods (O1 - O6). When clients of a class are not allowed to change the value of a characteristic directly and need to manipulate the characteristic through more complex mutators, the latter mutators must be implemented in terms of this setter. When there exists a default value for the characteristic then that value is always explicit added to the declaration, even if that value is the default value of the type of the internal representation. Thus, absence of a default value in the declaration means this characteristic must always be initialized during construction (O4). Figure 1 illustrates the internal representation with default value, basic query and setter for the characteristic balance. As the stored and observed values are equal the implementation of both methods is trivial. The basic query is annotated `@Raw` because we also want to be able to observe the state of the property balance when the object is not in a steady state.

```
private BigDecimal balance=BigDecimal.ZERO;

/**
 * Return the balance of this bank account
 */
@Basic @Raw
public BigDecimal getBalance(){
  return balance;
}

/**
 * Set the given balance as the balance of
 * this bank account
 * @post  new.getBalance() == balance
 */
public void setBalance(BigDecimal balance){
  this.balance = balance;
}
```

Fig. 1: Representation of the property balance

Value Requirements. For each property, *a Boolean inspector* is introduced to validate the value requirements. This inspector is the only place where these requirements are specified and implemented (O1 - O6). Because the result of this inspector is by definition independent of the state

```
/**
 * @return if ((creditLimit==null) ||
 *                 (creditLimit.signum() >= 0))
 *         then result == false
 */
public static boolean isProperValueForCreditLimit(
                  BigDecimal creditLimit){
  return  (creditLimit != null) &&
          (creditLimit.signum() < 0);
}

/**
 * @post  new.getCreditLimit() == creditLimit
 * @throws IllegalArgumentException
 *         !isProperValueForCreditLimit(creditLimit)
 */
public void setCreditLimit(BigDecimal creditLimit)
                  throws IllegalArgumentException{
  if (! isProperValueForCreditLimit(creditLimit))
    throw new IllegalArgumentException();
  this.creditLimit = creditLimit;
}
```

Fig. 2: Value Requirement of the property credit limit

of the object, the inspector is a class method (`static` in Java). By convention, the name of the inspector checking the VR for a property α is `isProperValueFor`α`(T` α`)` (O4, O6). According to P4, all business rules must be enforced in the application. Calling the setter with an actual argument that violates the VR is an exceptional situation and must be signaled. The setter is adapted accordingly. Figure 2 illustrates the inspector and setter for the characteristic credit limit. The specification of the inspector is worked out in a non-deterministic way. It specifies only which values are certainly not acceptable as value for the credit limit of a bank account. Notice however that the signature of the inspector `isProperValueForCreditLimit()` implies that only true or false can be returned as result. This way subclasses can decide to further restrict possible values or to explicitly confirm what values are always acceptable (O2, O5).

State Requirements. A state requirement describes a constraint that restricts acceptable value combinations of characteristics. Each SR is described by *a Boolean inspector*. This inspector is again the only place to specify and implement the SR at stake (O1 - O6). The inspector has an argument for each characteristic involved in the SR. Thus, this inspector is also a class method. Obviously, the value from each involved characteristic must meet the VR to have an acceptable combination of values. By convention, the name of a SR involving properties α and β is `isProper`$\alpha\beta$`(T1` α`, T2` β`)` (O4, O6). Each characteristic can be involved in multiple SR. We will illustrate in the paragraph about transition requirements how these inspectors are integrated in the setter. Figure 3 illustrates the SR between the properties balance and credit limit. The specification of this inspector is also non-deterministic; it is, however, also possible to close the specification and make it deterministic.

Invariant. The invariants for a class are described by the union of all VRs and SRs. We say that a characteristic α meets its invariants if it meets the VR and all the SRs it is involved in. For each characteristic α, we introduce *a Boolean inspector* to check whether a given value meets its invariants

```
/**
 * @return  if (!isProperValueForBalance(balance))
 *          then result == false
 * @return  if (!isProperValueForCreditLimit(
 *                         creditLimit))
 *          then result == false
 * @return  if (creditLimit.compareTo(balance)>0)
 *          then result == false
 */
public static boolean isProperBalanceCreditLimit(
    BigDecimal balance, BigDecimal creditLimit){
  return isProperValueForBalance(balance) &&
      isProperValueForCreditLimit(creditLimit) &&
      (creditLimit.compareTo(balance) <= 0);
}
```

Fig. 3: State Requirement between balance and credit limit

with respect to the current state of the object. By convention, the name of this inspector is `canHaveAs`α`(T` α`)`. As this method is the sum of the VR for α and all SRs where α is involved in, this method can be generated as a whole (O1, O4, O6). With respect to the property α, the object is in a steady state if the current registered value for α meets its invariants. The inspector `hasProper`α`()` specifies the invariant for α. This method can also be generated (O1, O4, O6). Figure

```
/**
 * @invar hasProperBalance()
 */
public class BankAccount {
...
/**
 * @return result==canHaveAsBalance(getBalance())
 */
@Raw
public final boolean hasProperBalance(){
  return canHaveAsBalance(getBalance());
}

/**
 * @return if (!isProperValueForBalance(balance))
 *         then result == false
 * @return if (!isProperBalanceCreditLimit(
 *                 balance, getCreditLimit()))
 *         then result == false
 */
@Raw
public boolean canHaveAsBalance(BigDecimal balance){
  return  isProperValueForBalance(balance) &&
          isProperBalanceCreditLimit(
              balance, getCreditLimit());
}
}
```

Fig. 4: Invariant from the property balance

4 illustrates these methods for the property balance. The inspector `canHaveAsBalance` is non-deterministic to allow new SRs in future subclasses (O2, O5). If new SRs are undesired the developer of this class can declare the inspector

`final` and make the specification deterministic. The inspector specifying the SR between balance and credit limit will be used in both the invariant inspector for balance and credit limit. By writing each SR in its own inspector, we avoid the need to duplicate that specification and implementation (O1 - O6). Both inspectors are annotated `@Raw`. Indeed, even when an object does not meet its invariants we want to be able to check if a given value meets its invariants.

Transition Requirements. A new value for a property must at least always meet the requirements described by the invariant. But often specific requirements restrict possible transitions when we take into account the current value of that property. The *Boolean inspector* canHaveAsNewα(T α) checks whether the given α is an acceptable new value with respect to the current state of the object (O2, O3, O4, O5, O6). First of all, the new value must meet its invariants. The extra TRs are added on top of them. The setter uses this inspector as guard for new values. Figure 5 illustrates this inspector and

```
/**
 * @return  if (!canHaveAsBalance(balance)
 *          then result == false
 * @return  let BigDecimal difference =
 *         getBalance().subtract(balance).abs() in
 *          result == difference.
 *                          compareTo(MAX_DELTA)<=0
 */
public boolean canHaveAsNewBalance(
                   BigDecimal balance){
  return  canHaveAsBalance(balance) &&
          (getBalance().subtract(balance).abs().
                   compareTo(MAX_DELTA)<=0);
}

/**
 * @post new.getBalance() == balance
 * @throws IllegalArgumentException
 *          !canHaveAsNewBalance(balance)
 */
public void setBalance(BigDecimal balance)
          throws IllegalArgumentException{
  if (!canHaveAsNewBalance(balance))
    throw new IllegalArgumentException();
  this.balance = balance;
}
```

Fig. 5: Transition requirement of the property balance

the adapted setter. Often a public setter will not be desired, mutators like `withdraw` and `deposit` are preferred above `setBalance`. It suffices to change the access modifier to `protected` (`private` doesn't allow subclasses to define custom mutators) and custom mutators can easily be specified in terms of this setter.

Construction. Construction is an event with very specific semantics. After the complete construction process an object must be in a steady state. Because that is also the first state of the object we don't have the compare the initial value of a characteristic with its previous value (there isn't one). Even when there is value assigned in the declaration to the instance variable, we don't consider that value as a 'previous' value. An immediate consequence is that we can't use the setter in the constructor. Because we still want to restrict the manipulating of the instance variable(s) to a single method we need to introduce a more basic setter: registerα(T α) (O1, O2). Figure 6 illustrates the basic setter for the property balance. Because this setter will be used in the constructor only the VR

```
/**
 * @post   new.getBalance() == balance
 * @throws IllegalArgumentException
 *      !isProperValueForBalance(balance)
 */
@Raw
protected void registerBalance(
                   BigDecimal balance)
          throws IllegalArgumentException{
  if (!isProperValueForBalance(balance))
    throw new IllegalArgumentException();
  this.balance = balance;
}
```

Fig. 6: Basic setter for the property balance

is checked in this setter. This setter is also necessary when we want to introduce a complex mutator that manipulates two via SRs related properties. The developer will have to build a custom transition checker for that mutator but that is a rather trivial task as all building blocks are available. Indeed, each VR and SR is specified in its own inspector (O1, O2, O5, O6).

A steady state after construction means that all VRs and SRs must be met. Unfortunately, we can not use the inspector canHaveAsα(T α) because this inspector assumes all other properties β, γ,... already have their value. As there is no order in the different assertions of the specification, using them is impossible. So, we are forced to repeat the invariant

```
/**
 * @effect   registerBalance(balance)
 * @effect   registerCreditLimit(limit)
 * @throws   IllegalArgumentException
 *      !isProperBalanceCreditLimit(balance,
 *                                  creditLimit)
 */
public BankAccount(
       BigDecimal balance,BigDecimal creditLimit)
                throws IllegalArgumentException{
  if (!isProperBalanceCreditLimit(balance,
                                creditLimit))
    throw new IllegalArgumentException();
  registerBalance(balance);
  registerCreditLimit(creditLimit);
}
```

Fig. 7: Construction of a bank account

conditions in the specification of the constructor. Fortunately, we can describe the semantics of the constructor in terms of other mutators, more in particular the basic setter, through the `@effect`-tag. This way we reduce the complexity of the specification and implementation (O1, O4, O6). So we only need to list all SRs in the `@throws`-clause. Figure 7 illustrates the constructor for the class of bank accounts.

Inheritance. On the one hand, a subclass can specialize a superclass. The subclass can adjust the semantics of inherited features. The Liskov Substitution Principle (LSP) [3] acts as a guideline to describe allowed adjustments. On the other hand a subclass can extend the superclass with new features. We will illustrate how the pattern copes with specialization and extension. A subclass may want to redefine the VR of a property. This means we need to be able to override the inspector checking the VR. Because the inspectors checking the VR are class methods and Java does not allow to override `static` methods the way a VR is implemented in the pattern needs to be adapted. Clearly, these inspectors need to be instance methods but on the other hand they have class semantics as their result is defined independent of the state of the object. Therefore, we move these methods to a static inner class. This static inner class implements the Singleton Pattern [2]: the object of the static inner class represents the outer class. The marker interface [4] `ClassObject` designates the static inner class. Figure 8 illustrates the inner class for the class of bank

```java
public class BankAccount {
  public static class COBankAccount
                              implements ClassObject{
  private static COBankAccount instance;

  protected COBankAccount(){}

  public static COBankAccount getInstance(){
    if (instance == null)
      instance = new COBankAccount();
    return instance;
  }

  public boolean isProperValueForBalance (...)
  {...}

  public boolean isProperValueForCreditLimit (...)
  {...}

  public boolean isProperBalanceCreditLimit (...)
  {...}
  }
}
```

Fig. 8: ClassObject inner class for the class BankAccount

accounts. The methods with class semantics can be moved without modification to the inner class. The specification and implementation of the instance inspectors using these methods can easily access them through the singleton object. A first advantage of moving the inspectors with class semantics into an inner class is that although they are instance methods can easily be identified as methods with class semantics (O4). A second advantage is that they make it impossible for the developer to use the state of the object erroneously (O6). A third advantage is that it is still possible to test these methods without needing an instance of the outer class (O3). If class B is a subclass of A, then the inner class of B must be a subclass of the inner class of A to be able to override methods from the inner class of A. Figure 9 illustrates the redefinition of the

```java
public class JuniorBankAccount
                              extends BankAccount{
  public static class COJuniorBankAccount
                              extends COBankAccount{
  /**
   * @return  if (!super.isProperValueForBalance(
                                        balance))
   *          then result == false
   * @return  if (balance.scale()!=0)
   *          then result == false
   */
  @Override
  public boolean isProperValueForBalance(
                              BigDecimal balance){
    if (!super.isProperValueForBalance(balance))
      return false;
    return balance.scale() == 0;
  }
  }
}
```

Fig. 9: Redefinition of the VR of the property balance

inspector checking the VR for the property balance. An extra constraint is added on top of the constraints defined in the class of bank accounts. The application, now, has two versions of the inspector checking the VR. The pattern must always use the right version. More in particular, the inspector must be invoked against the right ‘class object’. *Dynamic binding* ensures using the right version of an instance method. Therefore, an instance method is introduced to retrieve the right ‘class object’. Figure 10 illustrates how the right VR inspector is invoked through ‘dynamic binding’. Adding new properties to the subclass is

```java
public class BankAccount {
  public COBankAccount getClassObject(){
    return COBankAccount.getInstance();
  }

  public boolean canHaveAsBalance(
                              BigDecimal balance){
    return  getClassObject().
              isProperValueForBalance(balance) &&
            getClassObject().
              isProperBalanceCreditLimit(balance,
                        getCreditLimit());
  }
}

public class JuniorBankAccount extends ... {
  @Override
  public COJuniorBankAccount getClassObject(){
    return COJuniorBankAccount.getInstance();
  }
}
```

Fig. 10: ‘Dynamic binding’ of a ‘class method’

now straightforward. If a SR involves a property α from the superclass, the inspector `canHaveAs`α`(T` α`)` needs to be redefined at the level of the subclass. Figure 11 illustrates how the new SR between the properties balance and upper limit is added to the inspector checking the invariant constraints for balance. Lines 5-6 and 13-14 can be generated (O1). Figures 9 and 11 prove that redefinitions are easily developed (O2 - O5). VRs, SRs and TRs can independent of each other be redefined.

```
public class JuniorBankAccount extends ...{
/**
 *   @return if (!super.canHaveAsBalance(balance))
 *           then result == false;
 *   @return if (!isProperBalanceUpperLimit(balance,
 *                                 getUpperLimit()))
 *           then result == false
 */
@Raw @Override
public boolean canHaveAsBalance(BigDecimal balance){
  if (!super.canHaveAsBalance(balance))
    return false;
  return getClassObject().isProperBalanceUpperLimit(
                          balance, getUpperLimit());
}
}
```

Fig. 11: A SR involving the balance and the upper limit

Pattern skeleton. To summarize, Figure 12 shows a skeleton from the pattern for a property without specification. Given this generated code (O1, O6) the developer has only to (1) complete the inspector checking the VR (2) add an inspector for each SR in the inner class and extend the `canHaveAs`α to invoke the introduced inspector (3) complete the inspector checking the TR.

Language Construct. Figure 12 proves that an inherent problem with patterns is that it generates quite some boilerplate code. The need for patterns signals a lack of expressiveness of programming languages. Therefore, we present an extension to increase that expression power. Figures 13 and 14 illustrate how the example is completely worked out with a new language construct `Property`.

```
/**
 * The balance of this bank account
 * @Value balance != null
 * @State balance.compareTo(creditLimit) >= 0
 * @Trans balance.subtract(new.balance).
 *             abs().compareTo(MAX_DELTA) <= 0
 */
Property BigDecimal balance isRelatedWith
                                  creditLimit;

/**
 * The credit limit of this bank account
 * @Value creditLimit != null
 * @Value creditLimit.signum() < 0
 */
Property BigDecimal creditLimit isRelatedWith
                                  balance;
```

Fig. 13: The class of bank accounts

The importance of specification is upgraded, by making it an integral part of the construct. The specification describes the different kinds of requirements. They act as guards to validate values in an update operation. Three new tags are introduced to specify the semantics of a `property`, one for each kind of requirement we identified in section IV. The assertions used in the specification are Boolean expressions. (1) Each VR is preceded with a `@Value`-tag. A VR may be split over

```
public class Foo {

  public Foo(T α) throws IllegalArgumentException{
    registerα(α);
  }

  private T α;

  @Basic @Raw
  public T getα(){
    return α;
  }

  @Raw
  protected void registerα(T α)
                  throws IllegalArgumentException{
    if (!getClassObject().isProperValueForα(α))
      throw new IllegalArgumentException();
    this.α = α;
  }

  @Raw
  public boolean canHaveAsα(T α){
    if (!getClassObject().isProperValueForα(α))
      return false;
  }

  public boolean canHaveAsNewα(T α){
    if (!canHaveAsα(α))
      return false;
  }

  @Raw
  public final boolean hasProperα(){
    return canHaveAsα(getα());
  }

  public void setα(T α)
                  throws IllegalArgumentException{
    if (!canHaveAsNewα(α))
      throw new IllegalArgumentException();
    registerα(α);
  }

  public  COFoo getClassObject(){
    return COFoo();
  }

  public static class COFoo
                        implements ClassObject{
    private static COFoo instance;

    protected COFoo(){}

    public static COFoo getInstance(){
      if (instance == null)
        instance = new COFoo();
      return instance;
    }

    public boolean isProperValueForα(T α){
      return ...;
    }
  }
}
```

Fig. 12: The pattern for a property α

multiple tags. (2) Each SR is preceded by a `@State`-tag. Each property can be involved in an unlimited number of SRs. (3) Finally, a TR is preceded by a `@Trans`-tag. A SR is always symmetric, which means it applies equal to all properties involved. Despite of this symmetry, the specification doesn't need to be duplicated. Relations between properties need to be mentioned explicitly. The characteristics a property is related with are added to a list following the keyword `isRelatedWith` in the signature of the property. This implies that the specification of the semantics of a property can be spread over multiple properties. We don't consider this as a drawback though because to understand a requirement involving two properties, one has to understand the semantics of both properties anyway. This list also identifies clearly on which properties changes to the specification can have an impact on. By avoiding the duplication we fully support Parnas' principle [8] saying that each fact must be worked out in one, and only one, place. The specification is by definition non-deterministic. The semantics of an assertion Γ in a VR, SR or TR is:

```
if !(Γ)
then result == false
else result == Undefined
```

Thus, when the assertion Γ evaluates to false, the submitted value not acceptable. On the other hand, when the assertion

```
/**
 * The balance of this junior bank account
 * @Value balance.scale() == 0
 * @Trans !isBlocked
 */
@Override
Property BigDecimal balance isRelatedWith
           creditLimit,upperLimit,isBlocked;

/**
 * The credit limit of this junior bank account
 * @Value creditLimit.
 *        compareTo(new BigDecimal(-1000)) >= 0
 * @Value creditLimit.scale() == 0
 */
@Override
Property BigDecimal creditLimit isRelatedWith
                                 balance;

/**
 * The blocked state of this ...
 */
Property boolean isBlocked isRelatedWith
                                 balance;

/**
 * The upper limit of this bank account
 * @Value upperLimit >= 1000
 * @Value upperLimit <= 10000
 * @State balance.compareTo(
 *          new BigDecimal(upperLimit))<=0
 */
@Immutable
Property int upperLimit isRelatedWith
                                 balance;
```

Fig. 14: The class of junior bank accounts

evaluates to true the value may be acceptable. The semantics of the VRs of credit limit in Figure 13 is that non-effective positive or zero decimal numbers are certainly not a good value for a credit limit. Negative values *can* be good values. Subclasses are allowed to further specify the *open* part. The requirements specified in a subclass are *added* to the requirements specified in the superclass. The VR of the credit limit in the class of junior bank accounts for instance now specifies that only strictly negative integer numbers are acceptable values.

Evaluation. Up to now, the pattern has only been applied to academic problems. These experiments show that about 70% of the code for defining properties is boilerplate code. As an example the full definition of class of bank accounts counts 360 lines of Java code. About 250 of these lines are boilerplate code. The typical Java programmer is not tempted to write all these lines in original definitions of classes. In particular, he will not be eager to encapsulate the different kind of requirements in Boolean inspectors such as `isProperValueForBalance()`, `canHaveAsBalance()`, `canHaveAsNewBalance()`, etc. This either leads to duplicate code because the same requirement is repeated over and over again in different parts of the class definition, or it compromises adaptability in time and space. We therefore believe that more advanced language constructs are needed to introduce properties in classes. We still need to experiment with this pattern in the scope of industrial software systems. We expect the same results with respect to the mere definition of properties in such large systems. The pattern gives the programmer the opportunity to focus more on the business at stake.

VI. Related Work

The central idea of Model Driven Architecture (MDA) [11], [12] is to automate transformations between models. To enable this transformations the specification should be defined in a formal way. MDA uses Design by Contract (DBC) [13] to specify the semantics of models formally. DBC was developed by Bertrand Meyer as part of the Eiffel programming language [1], [18]. DBC is based amongst others on Hoare-logic [7] that already introduced concepts like *preconditions* and *postconditions*. Other object-oriented languages with native support for DBC are for instance Sather [20], Nice [19] and Spec# [21]. Commonly used languages like Java, C++ [14] and C# [22] have no support for DBC. However, several third-party tools have been developed for those languages. Tools for java are for example: Contract4J [23], JContractor [24]. The Java Modeling Language [25] is a behavioral interface specification language that can be used to specify the behavior of Java modules. B AMN [26] and UML-RSDS [27] present similar concepts. In UML-RSDS correct operations can be synthesized from invariants (VR and SR constraints in this paper) in many cases. In B, a TR can be expressed as an abstract pre-post specification which is correctly refined by a more concrete operation that ensures the TR constraints.

VII. Conclusion and Future Work

In Object-Oriented programming, a significant effort has been made in recent years to increase the expressiveness of programming constructs for the production of code. However, we have not seen such a similar evolution in the design and specification of code. Developers are often forced to write ad-hoc code specifications in a non-standardized manner. In this paper, we therefore took an evolutionary approach to language-integrated specification constructs. We started from existing specification constructs (@pre, @post, ...) with the ambition to enhance the overall expressiveness of specifications in object-oriented languages.

We have identified three types of requirements that can occur in program specifications: Value Requirements (VR), State Requirements (SR), and Transition Requirements (TR). **Value Requirements** are used to specify the most basic kind of business rules in that they restrict the range of values that a characteristic, property or association, can have. A **state requirement** describes a constraint that restricts acceptable value combinations of (a set of) properties. **Transition Requirements**, then, specify the business rules that restrict the evolution of property values.

Besides that, we feel that also need to help developers in correctly using these constructs. Therefore, we have introduced a "boilerplate pattern" that showcases the inspector methods required for validating the VR, SR and TR in a specification. But a pattern is, according to us, not sufficient as a solution, because (1) it involves too much boilerplate code and (2) there remains a risk of incorrectly implementing (a part of) the pattern, which would still lead to ill-defined specifications.

Therefore, we have integrated a Specification Language extension in Java. By means of the @value, @state and @trans tags, developers can better capture the Specification of their code, while outsourcing all technicalities to a code generator. We have also introduced the **isRelatedWith** construct in order to further minimize the risk of duplicate specifications. As an additional benefit, the formal specifications are compile-time checked, since they are injected in the Java code in the background, before compilation.

We recognize that this is the first step in our roadmap to develop a fully integrated, expressive Specification language, and would like to conclude by giving the reader a view of our upcoming research, which has two important future directions. Next to our Specification-to-Code generator, we also want to build a detailed formalization of the Specification language, in order to identify opportunities to further enhance the expressiveness of the concepts. A second direction is to further increase the expressiveness and action radius of the concepts. For instance, we are currently working to add determinism to our Specification constructs, for which a prototype definition is currently available, but too preliminary for this paper. Another example is that we are defining more finegrained rules on when properties may be added or removed to the isRelatedWith-list. These rules are currently still based on informal guidelines.

Acknowledgements We especially thank Sven De Labey for his advice. We thank the anonymous reviewers for their insightful comments. This research is funded by the the Fund for Scientic Research (FWO) in Flanders.

References

[1] B. Meyer, *Object-oriented software construction*, second edition ed., Prentice Hall, 1997.

[2] E. Gamma, and R. Helm, and R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*, Addison-Wesley Longman Publishing Co., Inc., 1995.

[3] B. H. Liskov and J. M. Wing, *A behavioral notion of subtyping*, ACM Trans. Program. Lang. Syst. 0164-0925 (1994), pp. 1811–1841.

[4] J. Bloch, *Effective java*, Java Series, Pearson Education, 2008.

[5] J. Bosch, *Design patterns as language constructs*, Journal of Object-Oriented Programming 11 (1998), pp. 18–32.

[6] M. Feathers, *Working effectively with legacy code*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.

[7] C. A. R. Hoare, *An axiomatic basis for computer programming*, Communications of the ACM 12 (1969), no. 10, pp. 576–580.

[8] D. L. Parnas, *On the criteria to be used in decomposing systems into modules*, Commun. ACM 15 (1972), no. 12, pp. 1053–1058.

[9] E. Steegmans, *Object oriented programming with java*, Acco, 2011.

[10] A.L. Rubinger and B. Burke, *Enterprise javabeans 3.1*, O'Reilly Media, 2010.

[11] D. Frankel, *Model driven architecture: Applying mda to enterprise computing*, OMG Press, Wiley, 2003.

[12] A. G. Kleppe, and J. Warmer, and W. Bast, *Mda explained: The model driven architecture: Practice and promise*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2003.

[13] R. Mitchell, and J. McKim, and B. Meyer, *Design by contract, by example*, Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 2002.

[14] S. B. Lippman, and J. Lajoie, and B. E. Moo, *C++ primer*, 5th edition ed., Addison-Wesley Professional, 2012.

[15] Javadoc Tool Home Page, *http://java.sun.com*, retrieved: 08, 2013

[16] Enterprise JavaBeans Technology, *http://java.sun.com*, retrieved: 08, 2013

[17] Eclipse project, *http://www.eclipse.org*, retrieved: 08, 2013

[18] Eiffel Software, *http://www.eiffel.com/*, retrieved: 08, 2013

[19] The Nice Programming Language, *http://nice.sourceforge.net*, retrieved: 08, 2013

[20] Sather, *http://www1.icsi.berkeley.edu/ sather*, retrieved: 08, 2013

[21] Microsoft Research Spec#, *http://research.microsoft.com/en-us/projects/specsharp*, retrieved: 08, 2013

[22] C# Programming Guide, *http://msdn.microsoft.com/en-us/library/vstudio/67ef8sbd.aspx*, retrieved: 08, 2013

[23] Contract4J, *http://www.polyglotprogramming.com/contract4j*, retrieved: 08, 2013

[24] M. Karaorman, and U. Holzle, and J. Bruno, *jcontractor: A reflective java library to support design by contract*, Tech. report, Santa Barbara, CA, USA, 1999.

[25] G. T. Leavens and Y. Cheon, *Design by Contract with JML*, http://www.eecs.ucf.edu/ leavens/JML//jmldbc.pdf, retrieved: 08, 2013

[26] J.-R. Abrial, *The B-book - assigning programs to meanings*, Cambridge University Press, 2005.

[27] K. Lano and S. K. Rahimi, *Synthesis of Software from Logical Constraints*, ICSOFT, 2012, pp. 355-358.

Applying Questionnaire to Assess the Lessons Learned Process in Software Project Management: a Case Study at GAIA

Marco Ikuro Hisatomi, Anderson de Souza Góes, and Rodolfo Miranda de Barros
Department of Computing
State University of Londrina (UEL)
Londrina, PR – Brazil
marco.hisatomi@gmail.com, andersonsouzagoes@gmail.com, and rodolfo@uel.br

Abstract — **In order to obtain benefits from the Lessons Learned Process in Software Project Management it is necessary to assess the process periodically. To avoid failures, assessments can be conducted based on questionnaires duly appropriate for each organisation or segment of the software project under development. Studies of Lessons Learned and Software Project Management have increased the assessments techniques and have guided the construction of assessment criteria in organisations. In this paper, we present a questionnaire template with different alternatives that offer different scores and axes of efficiency to enhance the assessment. We intend to demonstrate that this questionnaire template establishes parameters for accurate measurements of the assessment of the Lessons Learned Process.**

Keywords – Lessons Learned, Software Project Management, Assessment Questionnaire.

I. Introduction

Through research on the organisation itself and with the participation of the people involved in software development projects, it is possible to maintain an information database called Lessons Learned [1]. The content of this database is the result of activities performed within an organisation. Throughout the project, the experiences are accumulated in an organised way to form the Lessons Learned of the project team.

Structuring Lessons Learned Processes in an organisation is not always fast and there should be a constant discussion of the subject for all people involved. The benefits of Lessons Learned should be pursued by those involved in the development of a software project [2], in the life cycle of the software development process. This paper aims to identify the main points that can be improved for this organisation, either through classification of information, its source or characteristic, or by its complexity.

Aiming at the success of the project, the application of Lessons Learned in the organisation is one of the techniques that contribute to this goal [3]. The goals must be constantly pursued in spite of the diversity of resources involved, the complexities and restrictions required during the project. Therefore, in order to facilitate the decision during the project, the Lessons Learned is fundamental to promote assertiveness in these decisions.

The maintenance of Lessons Learned contributes positively to the successful delivery with the expected quality [3], even with the numerous innovative techniques in project management [4]. In this paper, the advantages offered by the Lessons Learned Process are proposed through the development of six axes of efficiency. The development of these axes intends to determine the level of efficiency of the maintenance of Lessons Learned Process that is being practised in the organisation.

By using the Lessons Learned Process in Software Project Management, organisations intend to guarantee effective collaboration in building the best software development techniques. Therefore, assessments should be conducted to measure the efficiency of this process. With that in mind, in this paper, we designed an assessment questionnaire to measure the efficiency of Lessons Learned for the main axes of the management process.

The present article is organised as follows: in Section 2, there is a literature review of the main bases of this project – Lessons Learned, Project Management and related work. In Section 3, we present the assessment based on the questionnaires at GAIA – Software Factory, as a case study. In Section 4, the process evaluation of Lessons Learned in Project Management. The results are presented in Section 5, and finally, the conclusion and future work are presented in Section 6.

II. Literature Review

We conducted [2] several studies for the literature review focusing on two areas: Project Management and Lessons Learned. The survey was developed from several documents with relevant and current issues in these areas. The study of related work also helped to consolidate the assessment proposed in this article.

A. *Lessons Learned*

The Lessons Learned Process includes organised activities for the recorded experiences of the people involved in a particular project and has great value as knowledge. Both positive and negative experiences are considered equally important in Lessons Learned, e.g., a variation of the technique of software testing can be positively considered; but if this variation results in failure, it can be considered as negative.

In [5], a Lesson Learned is considered as so when it has an impact on daily operations. Basically, adverse experiences are observed and used to improve the organisation or a particular member of staff. In all cases, the result should be, among others, a significant reduction of effort, an improvement in design, and an optimisation of computer resources.

Among the several applications of Lessons Learned, with beneficial impacts to the organisation, some can be cited according to [3]:

- Time saved in solving problems, since the solutions of common problems are centralised in one location for easy access by members.
- Reduction or elimination of costs from avoiding the same work to be done again when correcting discovered defects.
- Encouragement of the use of best practices within the organisation, which increases the likelihood of success of the projects.

Narratives that explicit knowledge or understanding gained through experience – both positive and negative – can still be characterised as a lesson learned. The lesson relates what was expected to happen, the facts and deviations that happened, the analysis of the causes of these deviations and what might be learned during the process [6].

The record of the lessons learned is an excellent way to avoid the same mistakes previously made and to replicate the successes achieved in the past to future projects. According to [7], five points are listed for a successful implementation of the process of documentation of the lessons learned:

1) Training of members of the organisation – it is necessary to change the paradigm that the collection and recording of lessons learned is a waste of time, and to bring to knowledge the benefits that information sharing has in an organisation. For this process to work, it is very important that the manager is able to generate motivation and involvement of all. According to [8], to make full use of the practices of knowledge management in a company, one of the key factors is the involvement of both stakeholders and workforce – which involves a change of habits.

2) Collection and recording of experiences – this task is considered to be costly and demands great effort from the staff. This task should be performed using practices and oriented towards an easy method of items relevant for the organisation; also, it is important that these items are organised following a set pattern.

3) Analysis of successes and failures – it is not enough that the lessons learned are simply recorded and catalogued; they also must be understood and analysed. After the identification of the activities that resulted in good results or failures, these records must be part of the knowledge basis. In that way, the Lessons Learned Process becomes an opportunity for analysing facts and for adopting measures for a continuous improvement.

4) Dissemination of knowledge – Simply archiving these lessons is not enough; they should also be disclosed throughout the organisation. This disclosure must take into account the direction and prioritisation of such information in accordance with the interests of each group.

5) Updated records – It is very important to understand that the register of the Lessons Learned should be cyclical, i.e., it must be constantly updated.

B. Project Management

Software development has been one of the major technological advances of our days, in the information age. All products built based on projects have shown positive results and measurable improvements in the future [4]. For that, project management is an activity largely applied to software development, which has improved significantly with less effort.

The most widely accepted definition for the term "project" is presented by the Project Management Body of Knowledge (PMBOK), and characterised as "a temporary endeavour undertaken to create a product, service or result only". In the same line, [9] defines it as "a unique venture to produce a set of results according to constraints of time, cost and quality clearly defined".

The great amount of software projects in progress, the number of people involved in these tasks and the tight delivery deadlines increases the complexity of these projects [10]. Therefore, there is a growing practice of Project Management (PM) for new software projects, whether new products or changes in systems already developed. According to [11], PM is used by organisations to manage the innovations in their processes. Thus, encouraging the creation and dissemination of organisational management techniques in these organisations is fundamental to improve products and processes services.

According to [12], there is the PM-specific branch of the organisation’s activity, because it includes various techniques in different business areas, such as: general administration, military organisation and engineering, among others. The activities involved in PM are multidisciplinary and require a lot of expertise and the participants' commitment to its implementation. The growth of project management refers to topics such as roles and responsibilities, organisational structures, delegation of authority, decision-making and especially corporate profitability [13].

Thus, the project management "is the application of knowledge, skills, tools and techniques to project activities to meet project requirements". Once the characteristics of the delivery products (and services) [4] [9] are defined, the activities must meet these objectives in an explicit – and not implicit – way. Throughout their development, the projects are organised according to their life cycle and divided in two classes: the projects that involve the activities of PM and the products that include the activities of product development [12].

During the development of a product, the tasks may vary according to the branch of industry (software, pharmaceutical, manufacturing, etc.), while the PM is independent from the segment. They can be classified into groups (called stages), such as initiation, definition, planning, execution, controlling, and closing. Each stage brings together activities with similar purposes, but with their own features and goals.

C. Related Work Process and Lessons Learned

The Process Management of the Lessons Learned is increasing, especially in the area of Information Technology, aiming at consolidating this process in software projects. For example, [14] proposes an architectural model for managing Lessons Learned in the testing phase. Although there are studies reporting the importance of this process, none of them includes the assessment questionnaire.

In the work of [15], the authors developed a guide containing major errors in the Lessons Learned. This subject was widely discussed by National Aeronautics and Space Administration (NASA) especially after the incident with the space shuttle Columbia. These authors' proposal is to consider the following key steps: the collection of lessons learned, their management and application in future projects.

According to [16], Lessons Learned Process is used to develop and maintain an organisational memory for a technology centre that develops high risk systems. In this centre, through interviews, decomposition and reintegration of tacit knowledge with explicit information, including the information gathering and dissemination, they managed to establish a process and obtained good results after its implementation.

In software engineering, the process of knowledge dissemination is based on Lessons Learned [17] in order to maintain a community of interest. This work describes the operation of the engineering centre software based on Commercial Off-The-Shelf (COTS) and how the Lessons Learned is used. It also includes a detailed description of a repository Lessons Learned and a planning evolution for it.

As previously mentioned, the use of Lessons Learned Process has clear aims in project management, which in turn requires the assessment of this process so that improvements can take place. Several studies focus on Lessons Learned Process, highlighting its necessity and advantages. However, the formal and effective assessment of this process is not always correctly explained.

In this paper, a formal assessment with the aim of improving Lessons Learned Process is proposed through a questionnaire applied to all involved in the project. Based on the results of this questionnaire, the organisation can decide how to employ its resources for each level of need indicated by the axes of efficiency.

III. ASSESSMENT OF LESSONS LEARNED PROCESS BASED ON A QUESTIONNAIRE

There are several methods of gathering information to meet an internal process in the organisation. In [18], one of them includes the collaboration of the team members that participated in this survey, so the accuracy of the information given is associated to the participants' commitment. Following this principle, a questionnaire was developed to be applied to all members of the software development team.

An assessment was conducted, as a case study, at GAIA Software Factory of the Department of Computing, State University of Londrina. This organisation was chosen because it develops software of various scopes for the university itself and, specially, because it is formed by undergraduate and postgraduate students that – after their graduation – leave the Factory, leading to knowledge loss. Other important factors were listed in this case study:

- An environment focused on software development;
- The team works with procedures, attributes and templates that can be reused;
- An organisation focused on a continuous improvement of its processes;
- The development software process includes integration at several software engineering area and governance in Information Technology (IT);
- Specialists in knowledge management;
- Experienced staff in project management.

In the work of [19], a method was used for multiple-item development of a questionnaire. The main objective of this method was the measurement of a universe through issues that represent reality. Alternatives distributed between strong, weak and staggered tend to result in accuracy. Thus, the questionnaire represents a powerful tool for the measurement of a situation.

In order to obtain a broad and complete picture, the questionnaire was built with objective questions, containing qualitative and quantitative alternatives. Beyond the issues that were considered in the axes of efficiency that lead to good practices for each process involved in the use of Lessons Learned Process. The axes of efficiency show trends for each of the items considered [20].

As illustrated in Figure 1, these axes represent the main features to use effectively the resources from Lessons Learned in a Software Development organisation. Through these axes of efficiency it is possible to focus on a specific feature, facilitating the management of Lessons Learned Process.

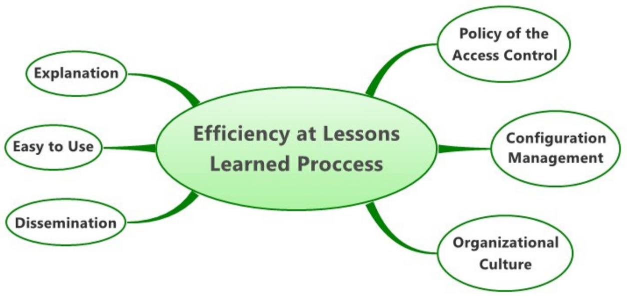


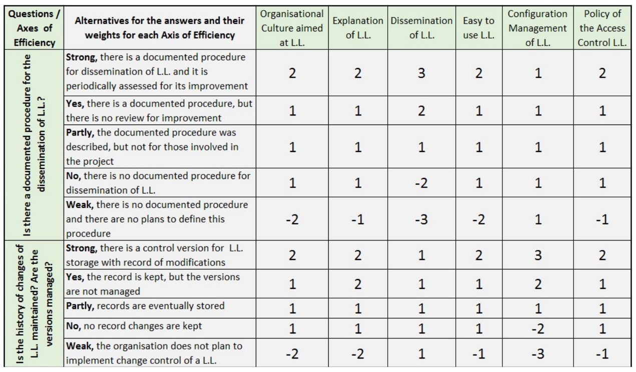
Figure 1. Axes of efficiency in Lessons Learned Process (adapted from [20])

Each question was elaborated with the objective of indicating the need of improvement. Also, it was considered getting a diagnosis for the applicability of the process related to the efficiency of Lessons Learned Process, according to the axes of efficiency. Each alternative indicates the level of this applicability, which can be achieved by the weight associated to each axis of efficiency, due to the impact that this response will provide to the axis.

The weight of the efficiency as a function of the sum is generally a multiplication, with values ranging from 0 to 3, in which 0 represents 'no influence', 1 'low influence', 2 'medium influence' and 3 'high influence'. To this levels of influence is added either the signs (+) or (–) determining, respectively, a positive or negative influence. The alternatives suggest that the participant will be framed according to their degree of participation in the Lessons

Learned Process. As an example, Figure 2 shows the possible alternatives for the two issues.

In Figure 2, the answer to each question consists in an alternative, which will be chosen by those involved in the development project. In the questionnaire, the participant answers the questions without knowing neither the correspondent weights nor the axes of efficiency. However, for each answer, it will be computed in a general sum and the representativeness of a participant's answer will be given by their answer multiplied by its respective weight.



Questions / Axes of Efficiency	Alternatives for the answers and their weights for each Axis of Efficiency	Organisational Culture aimed at L.L.	Explanation of L.L.	Dissemination of L.L.	Easy to use L.L.	Configuration Management of L.L.	Policy of the Access Control L.L.
Is there a documented procedure for the dissemination of L.L.?	**Strong,** there is a documented procedure for dissemination of L.L. and it is periodically assessed for its improvement	2	2	3	2	1	2
	Yes, there is a documented procedure, but there is no review for improvement	1	1	2	1	1	1
	Partly, the documented procedure was described, but not for those involved in the project	1	1	1	1	1	1
	No, there is no documented procedure for dissemination of L.L.	1	1	-2	1	1	1
	Weak, there is no documented procedure and there are no plans to define this procedure	-2	-1	-3	-2	1	-1
Is the history of changes of L.L. maintained? Are the versions managed?	**Strong,** there is a control version for L.L. storage with record of modifications	2	2	1	2	3	2
	Yes, the record is kept, but the versions are not managed	1	2	1	1	2	1
	Partly, records are eventually stored	1	1	1	1	1	1
	No, no record changes are kept	1	1	1	1	-2	1
	Weak, the organisation does not plan to implement change control of a L.L.	-2	-2	1	-1	-3	-1

Figure 2. Issues, alternatives, effiency and weights (Produced by the authors)

For example, the question "Is there a documented procedure for the dissemination of L.L. (Lesson Learned)?" has five alternatives and each one represents gradually the position of the organisation. It is possible that the weights were assigned to each alternative along an axis of efficiency. In spite of the fact that the weights of each answer have a simple score calculation, this format leads to results more accurately dependent on the granularity of each alternative.

Different weights were attributed for the axis "Dissemination of Lessons Learned": '3' to the alternative "Strong, there is a documented procedure for dissemination of L.L. and it is periodically assessed for its improvement", which indicates a high positive impact to this issue. And so on, up to the weight '–3' for the option "Weak, there is no documented procedure and there are no plans to define this procedure", indicating that there is a high negative impact.

IV. Assessment of Lessons Learned Process in PM

Based on the proposal and on the issues raised and described in Figure 2, we conducted a case study about the advantages that its use can provide to the application of the Lessons Learned Process as part of software development project management. Then, we explain three main advantages of the questionnaire with axes of efficiency in the assessment of the Lessons Learned Process ([1], [3], [14], [21]). These themes were established according to the studies conducted in the organisational environment of systems development, in which there is collaboration for project management.

The benefits of each axis of efficiency lie on the results that each of them will provide for the development of systems and especially for the process of software development with Lessons Learned Process. For a positive result in project management it is necessary that the analysis [22] for each axis be part of the process. Thus, each axis of efficiency will demonstrate the contribution to the whole process.

A. *Explanation of Lessons Learned*

Knowledge is valid for people when the development of a task can be controlled by them, and adapted to specific needs, i.e., when it becomes a Lesson Learned. That, in turn, may or may not be spelled out for future use or shared with others [23]. Sharing a Lesson Learned with a software project development team becomes an advantage that can result in minimisation of effort or improvement of the final product.

The possibility of having an organisational integration of knowledge management and, as a consequence, gaining a competitive advantage in the market [21], represents the importance of explaining Lessons Learned. When described, a Lesson Learned becomes reference for use, association or improvement of a given process or task within the organisation.

In order to complete the cycle of information, according to [23], knowledge must transit between tacit and explicit, in phases of socialisation, externalisation, combination and internalisation. On the other hand, [24] states that before the storage of information, knowledge must be made explicit, classified and integrated, so it can contribute to improvements and add new information. Thus, the axis of Explanation of the Lessons Learned is considered essential to a formal process; without which it would be impossible to continue the treatment and use of acquired knowledge.

B. *Ease of search*

The models and materials surveyed do not offer an explicit description of the minimum criteria that would facilitate the recovery of the Lessons Learned stored. Capability Maturity Model Integration (CMMI) reports that the recovery of a Lesson Learned should have criteria to facilitate this process; however, it does not show how this procedure should be done.

The tool used to storage the Lessons Learned facilitates the search and access, encouraging the practice of the process as a whole [20]. Besides that, the design is optimised in time, considering that several people access several times the knowledge repository. If for each survey the time can be optimised, then the time of the task can also be reduced.

The easy access to a Lesson Learned is a major factor that drives the effective use of this process. Since a Lesson Learned is explicit, it should be made available in a simple way for the consultation process. Some keywords are fundamental to enable this search by those involved in the project. Still, according to [24], this question must have the correct rating to have the assertiveness and adequate categorisation also highlighted by [27]. With the combination of these practices, naturally, one can predict that the good explanation and ease of use will boost the dissemination of Lessons Learned.

C. *Configuration Management*

One of the most important activities in software development is the responsibility of the project manager to control revisions and versions. In this process, all changes are controlled in an organised and predictable way [26], which can be envisaged for specific versions for each project phase. The great advantage in the use of configuration management is the control of the record of Lessons Learned, for a more efficient decision-making. Comparing a Lesson Learned at the exact moment in which its use is being analysed [16] ensures the manager an effective decision for its application or not.

This article aims to contribute to the studies analysed with an approach in an appropriate format and customisable for the assessment of Lessons Learned Process at a software development organisation. The axis of efficiency Configuration Management will complement the stability to use the Lessons Learned Process, along with the other axes. The collaboration of this axis is directly related to the axes of Policy of the Access Control and the Dissemination of Knowledge. Through proper configuration of each item a Lesson Learned will enable the correct version of the item according to the responsibility of the project member giving, therefore, greater security and reliability of the information to the project manager.

V. RESULTS

The organisation's choice to conduct the questionnaire was crucial because the GAIA has development model components that are organised and trained to act in this segment. In GAIA, there are three groups: specialists, non-specialists, and project manager. In this way, it becomes feasible to conduct the questionnaire, and because all are involved, daily, in the development software process.

Following the process, proposed by [21], as a methodology for collecting results, research was targeted at the three groups of the software development team in order to obtain the recognition of the assessment questionnaire proposed by this article. The team involved and engaged belongs to the project GAIA – a Software Factory, a research and extension project of the Department of Computing, State University of Londrina (UEL). People were classified into three groups: project manager, specialists, and non-specialists. In the first group, there were those who know and perform project management; in the second, experts of Knowledge Management; and in the third group, the other participants of the software development team.

The statements were designed in accordance with the suggestions of the participants of the software development team, based on guidelines found in the literature ([21], [25]). The statements, presented in Table I, are essential to the targeting of objectives of knowledge management and assessment of Lessons Learned Process. In order to prepare Table I with the statements, we initially prepared questions about GAIA's Lessons Learned and we conducted it to the three assessment groups. The questionnaire was designed considering the axes of efficiency, treated in Section 3, for a more effective assessment.

TABLE I. LIST OF THE ASSESSORS' CLAIMS

No.	Statements
01	Lessons Learned Process is initiated by explicit knowledge, supporting the possibility for use and improvements.
02	With the management of Lessons Learned participants will have greater confidence to use and work with the repository.
03	Lessons Learned explained and managed becomes part of the organisation and is not restrict to the expert who wrote it anymore.
04	Participants will have the greatest stimulus in the use of Lessons Learned when it is easy to use.
05	Saving versions of changes in Lessons Learned will enable checking the evolution of its use.
06	Lessons Learned is not available without access control, which will maintain its integrity in a controlled way.
07	It is possible to measure the efficiency of use from the reach of a Lesson Learned to the ones involved in the project.
08	The questionnaire will direct the evolution of the quality of the process of Lessons Learned.
09	Through the questionnaire containing the axes of efficiency, the margin of safety in the assessment of the Lessons Learned Proces will be increased.
10	The weights allocated to each of the answers indicate greater importance in the application of Lessons Learned.

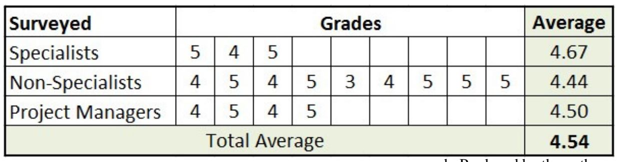
a. Produced by the authors.

In Table I, we present the advantages of completing the assessment of the Lessons Learned framework with the axes of efficiency. The efficiency of the proposed process is depicted in Table II, which shows a positive assessment, with scores close to the maximum. The great advantage – presented in this paper – of the assessment methodology using the questionnaire is specifically the objectivity to validate each point represented by the axes, the contribution to the effectiveness and efficiency in project management in a software development organisation.

The applicability of the questionnaire for assessing the Lessons Learned Process was presented and submitted in the organisation model of software development GAIA, among the three groups. Based on the model presented in Figure 2, the questionnaire aimed to assess whether it is possible to have positive results for assessment of Lessons Learned Process in a software development organisation.

Each group – specialist, non-specialist and project manager – analysed the problems with their weights, responding according to the applicability in the development environment GAIA. Data were collected after each participant gave a score between 1 and 5, in which 1 means 'strongly disagree', 2 means 'partially disagree', 3 means 'agree', 4 means 'partially agree' and 5, 'fully agree', consistent with the proposed assessment. Finally, the optimistic result is expressed in Table II.

TABLE II. TABULATION OF SCORES OF ASSESSORS



Surveyed	Grades									Average
Specialists	5	4	5							4.67
Non-Specialists	4	5	4	5	3	4	5	5	5	4.44
Project Managers	4	5	4	5						4.50
Total Average										**4.54**

b. Produced by the authors.

The assessment of this work is very satisfactory, since the percentage among the assessors numbered 90.8% (4.54/5.00) at the final average, as demonstrated in Table II, ensuring effective use of the questionnaire. For a good Lessons Learned Process management, it is possible that a software development company measure through the questionnaire process, with axes of efficiency and weights for each alternative of the question.

In this summary, we have demonstrated that 50% of project managers pointed favourably for the use of this methodology, giving evidence that the results are satisfactory for the Lessons Learned Process in software development. With this favourable outcome, the process becomes an opportunity for improvement in software development, boosting the possibility of investing in the assessment and improvement of Lessons Learned Processes.

VI. CONCLUSIONS

The main objective in the Lessons Learned Process will be achieved in a collaborative way among project participants when sharing experiences about several criteria that are being met. It will not be possible to achieve a positive benefit of a Lesson Learned unless the different parts of the process are seen as complementary.

Although methods for assessing a process aim at measuring the efficiency and effectiveness within a determined period of time, a survey conducted using the questionnaire has the advantage of obtaining accurate results with a specialisation according to the organisation. Depending on the questionnaire, whether it is prepared with the axes of efficiency and alternatives on scales, the results may be precise and reliable.

In this article, it is evident that the interrelationship among the axes of efficiency, with the intention of Lessons Learned, will be essential to this process. To ensure that the axes are kept in balance, there is a need to plan and execute a periodic assessment [5]. The ongoing assessment will ensure that all improvements will be identify to maintain the balance among the axes of efficiency aiming at maintaining the Lessons Learned Process. If one of the axes suffers greater positive change, the other must also be revised so that they are all levelled.

Much of the evolution of these processes – including Lessons Learned – is improved from constant assessment. According to [28], conducting questionnaires in assessment processes has advantages for both the staff and the process. Likewise, this advantage was verified by the assessment of team project GAIA, mainly by specialists and project managers, certifying that this questionnaire model is valid to maintain and improve the Lessons Learned Process in software development organisations.

Without the questionnaire it would be impossible to identify the possible need for changes in the Lessons Learned Process. Each member who answered the questionnaire can review how the Lessons Learned Process can impact the quality of software development. When answering the questionnaire, the member of the development team reflects on their performance before the alternatives of the questions. This reflection is notice in accordance with the objective of pushing the Lessons Learned Process.

Following the good results of the questionnaire model we envisage, in a future work, the application of the same model in private companies in the North region of the State of Paraná. This region has several technology companies, with the potential of producing high quality software and diversity training for more than three hundred professional graduates every year. The application of this model in organisations of various industries will demonstrate that the results are significant for more secure conclusions on the use of Lessons Learned within this market segment.

REFERENCES

[1] A. de Souza Goes and R. M. de Barros, "Gerenciamento do conhecimento em uma fábrica de software: Um estudo de caso aplicando a ferramenta GAIA – L.A.", In Conferencia Latinoamericana en Informática (CLEI), 2012.

[2] A. de Souza Goes, M. I. Hisatomi, B. M. Omena, and R. M. de Barros,, "Applying Lessons Learned as an Improved Methodology for Softtware Project Management", International Conference Information Systems (IADIS), 2013, pp 302-306.

[3] T. H. Roe, "Establishing a Lessons Learned Program: Observation, Insights and Lessons", Center for Army Lessons Learned: USA, 2011, pp. 88.

[4] PMBOK, "Um Guia do Conhecimento em Gerenciamento de Projetos", Quarta Edição, Project Management Institute, Inc, Newtown Square, Pennsylvania EUA, 2008.

[5] MPS-BR (Lições Aprendidas), "Associação para Promoção da Excelência do Software Brasileiro"; "Melhoria de Processo do Software Brasileiro Guia Geral Sumário", SOFTEX, 56p., 2011.

[6] B. H. Reich, A. Gemino, and C. Sauer, "Knowledge management and project-based knowledge in it projects: A model and preliminary empirical results", International Journal of Project Management, 2012, pp. 663-674.

[7] M. G. Aldenucci, "Um modelo de maturidade para o processo de gerenciamento de riscos", Dissertação de Mestrado, Pontifícia Universidade Católica: PUC-PR, 2009.

[8] K. F. Brett and D. Dressler, "The 24 Keys to high performance", Frontline Group Organizational Learning Division, 2000.

[9] J. Westland, "The Project Management Life Cycle: Complete Step by Step Methodology for Initiating, Planning, Executing & Closing a Project Successfully", Philadelphia, PA: Kogan Page, 2006.

[10] P. C. Torreão, "Ambiente Inteligente de Aprendizado para Educação em Gerenciamento de Projetos". Dissertação de mestrado, Universidade Federal de Pernambuco. Recife: UFPE, 2005.

[11] L. V. Martins, "Gestão Profissional de Projetos". Revista TecHoje. Belo Horizonte: ITEC, 2003.

[12] P. C. Dinsmore and J. Cabanisbrewin, "The AMA Handbook of Project Management", Second Edition. New York: AMACOM, 2006.

[13] H. Kerzner, "Project Management: A systems Approach to Planning, Scheduling and Controlling", Eighth Edition. Hoboken, New Jersey: John Wiley & Sons, 2003, pp. 211.

[14] J. Andrade, J. Ares, M.-A. Martinez, J. Pazos, S. Rodríguez, J. Romera, and S. Suárez, "An architectural model for software testing lesson learned systems, Information and Software Technology", vol. 55, n. 1, 2013, pp. 18-34.

[15] E. W. Rogers, R. L. Dillon, and C. H. Tinsley, “Avoiding Common Pitfalls in Lessons Learned Processes that Support Decisions with Significant Risks”, Aerospace Conference, 2007, pp. 1-7.

[16] D. Mendoza and R. Johnson, “Using a Lessons Learned Process to Develop and Maintain Institutional Memory and Intelligence”, Aerospace Conference, 2006, pp. 1-10.

[17] I. Rus, M. Lindvall, C. Seaman, and V. Basili, “Packaging and Disseminating Lessons Learned from COTS-Based Software Development”, Proceedings of the 27 th Annual NASA Goddard/IEEE Software Engineering Workshop, 2003, pp. 131-138.

[18] H. Günther, “Como Elaborar um Questionário”, 2003, pp. 01-14.

[19] Y.-Y. Chen and H.-L. Huang, “Knowledge management fit and its implications for business performance: A profile deviation analysis”, Knowledge-Based Systems, 2012, pp. 262-270.

[20] G. U. Briganó, “Um framework para desenvolvimento de governança de TIC. 2012. 155. Dissertação de Mestrado em Ciência da Computação – Universidade Estadual de Londrina, Londrina, 2012, pp. 18, 29-55.

[21] S. Rautenberg, A. V. Steil, and J. L. Todesco, “Modelo de Conhecimento para mapeamento de instrumentos da gestão do conhecimento e de agentes computacionais da engenharia do conhecimento” Perspectivas em Ciência da Informação, v.16, n.3, 2011, pp. 26-46.

[22] P. Carrillo, K. Ruikar, and P. Fuller, “When will we learn? Improving lessons learned practice in construction”, International Journal of Project Management, 2013, pp. 567-578.

[23] I. Nonaka, R. Toyama, and N. Konno, “SECI, Ba and Leadership: a Unified Model of Dynamic Knowledge Creation”, Leadership, vol. 33, 2000, pp. 5-34.

[24] J. Xue and Z. Zhang, “The Research on the Application Strategies of Information and Communication Technologies to Promote the Knowledge Transfer in Regional Innovation System”, 2006, pp. 138-145.

[25] B. Cakici and M. Boman, “A workflow for software development within computational epidemiology”, Journal of Computational Science, 2011, pp. 216-222.

[26] ITIL Version 3 Service Transition, pp. 65 – 68.

[27] E. Serna M., “Maturity model of Knowledge Management in the interpretativist perspective”, International Journal of Information Management, 2012, pp. 365-371.

[28] F. E. A. Horita, M. I. Hisatomi, F. H. Gaffo, and R. M. de Barros, “Maturity Model and Lesson Learned for improve the Quality of Organizational Knowledge and Human Resources Management in Software Development”, International Journal os Software Engineering and Knowledge Engineering, 2013, pp. 552-555.

Refactoring to Static Roles

Fernando Barbosa
Escola Superior de Tecnologia
Instituto Politécnico de Castelo Branco
Castelo Branco, Portugal
fsergio@ipcb.pt

Ademar Aguiar
INESC TEC and Departamento Informática
Faculdade de Engenharia da Universidade do Porto
Porto, Portugal
ademar.aguiar@fe.up.pt

Abstract— **Roles can be used to overcome some composition limitations in Object Oriented Languages and contribute to a better code reuse, reducing code replication and improve code maintenance. Therefore, the refactoring of legacy code to roles is an important step in maintaining and evolving this code. In this paper, we present refactorings to convert a system to roles We also present some refactorings that enable roles to be even more reusable.**

Keywords- roles; refactoring; code reuse; code maintenance

I. INTRODUCTION

The "tyranny of the dominant decomposition" states that a single decomposition strategy cannot capture all possible views of a system [1], so there are always concerns that cannot be adequately decomposed and are scattered among the various modules. Several decomposition alternatives have been proposed: mixins [2], traits [3], features [4], aspects [5] and both dynamic [6][7] and static roles [8].

We use static roles as defined by Riehle in [6]. We do not use roles as dynamic entities that can be attached or detached from objects. There is much work on dynamic roles [7][9] [10] so this is mentioned to avoid confusion. Our static roles model concerns that are a subset of a class responsibility: those that are not the class main concept. Roles compose classes by adding their code to the class. The class interface can be seen as a whole or as a union of all the methods offered by the roles it plays. To program with roles we use JavaStage, an extension to Java. For more information on static roles and JavaStage we refer to [8].

Our experience with the use of static roles showed that they provide better decomposition when compared to class decomposition [8]. We would improve legacy systems if we make them use roles. The use of roles would provide a better way to reuse code, eliminate code replication, enhance the systems' modularization and easy maintenance.

Refactoring [11][12] is program transformation where the program maintains its behavior but is improved in non-functional qualities like readability, reuse or changeability. We can use refactorings as a way to transform a system without roles into a system with roles. There is not, however, a catalogue for role related refactorings. To fill this gap we present, in this paper, a collection of role related refactorings.

In these refactorings we use the JavaStage language [8] because it is backward compatible with Java and JVM compliant. Existing systems can be upgraded to roles in a transparent way to their users.

The refactorings were developed using our experience using roles to reduce code replication in several systems, including those referred in [8], and also when developing design patterns using roles [13]. The proposed refactorings may not be complete but they provide a starting point for a role refactoring catalogue. Our experience has been transforming existing systems into roles and not developing and maintaining systems with roles, so there may be some refactorings that only deal with roles yet to be discovered.

The rest of the paper is organized as follows: Section II shows a refactoring example. Section III presents the advantages of refactoring to roles. Section IV presents the proposed refactorings. Section V deals with related work and Section VI concludes the paper.

II. A FIRST EXAMPLE

Consider the example of Figure 1 which shows an excerpt of an Abstract Figure class that is a superclass for all the figures in a drawing application. The figure must warn the view whenever it is changed so the view can be updated. An Observer [14] is used for this purpose. We can argue that being a subject is not the class's main concern. From this we can say that the code from lines 7 to 20 should not be in the class. We can put that code into a FigureSubject role by using Extract Role. The outcome is shown in Figure 2.

A role may define methods and fields with access levels (lines 11-26). To play a role the class uses a plays directive and gives the role an identity (line 2). A class playing a role is called a player of the role. When a class plays a role all the non private methods of the role are added to the class.

Looking at the role we can see that to use it in other situations we could just use another observer type. Ignoring methods names, for now, we could apply the Replace Type with Generic refactoring and build the role in Figure 3.

We can observe that what prevents this role from being reusable for other instances of the observer pattern are the methods names. The methods that require the use of Make Method Name Configurable are the methods that add and remove observers, the fire methods and the update methods.

The JavaStage language allows the configuration of a method name. It can also require certain collaborators to have specific methods. These features are used in the Make Method Name Configurable refactoring.

Each configurable method name may have three parts: a configurable one and two fixed (optional). The configurable part is bounded by # as in fixed#config#fixed. Configuration is done by the class playing the role in the plays clause.

```
public class AbstractFigure implements Figure {
  private Color color;
  public void moveBy(int dx, int dy) {
     fireFigureMoved( );  }
  public void setColor( Color c ){
     fireFigurePropertyChanged( );   }
  private Vector<FigureObserver> observers =
                  new Vector<FigureObserver>();
  void addFigureObserver(FigureObserver o){
     observers.add( o ); }
  void removeFigureObserver(FigureObserver o){
     observers.remove(o);}
  protected void fireFigureMoved( ){
     for( FigureObserver o : observers )
        o.figureMoved( );
  }
  protected void fireFigurePropertyChanged( ){
     for( FigureObserver o: observers )
        o.figurePropertyChanged( );
  }
}
```

Figure 1 An excerpt of an AbstractFigure class doing work outside its main concern

```
public class AbstractFigure implements Figure {
  plays FigureSubject figSubject;
  private Color color;
  public void moveBy(int dx, int dy) {
     fireFigureMoved( );
  }
  public void setColor( Color c ){
     fireFigurePropertyChanged( );
  }
}
public role FigureSubject {
  private Vector<FigureObserver> observers =
                  new Vector<FigureObserver>();
  void addFigureObserver(FigureObserver o){
     observers.add( o ); }
  void removeFigureObserver(FigureObserver o){
     observers.remove(o); }
  protected void fireFigureMoved( ){
     for( FigureObserver o : observers )
        o.figureMoved( );
  }
  protected void fireFigurePropertyChanged( ){
     for( FigureObserver o : observers )
        o.figurePropertyChanged( );
  }
}
```

Figure 2 The class from Figure 1 now refactored to roles.

```
role Subject<ObserverType> {
  private Vector<ObserverType> observers =
                  new Vector<ObserverType>();
  void addObserver(ObserverType o){
     observers.add( o ); }
  void removeObserver(ObserverType o){
     observers.remove(o); }
  // ...
}
```

Figure 3 The role from Figure 2 using other types of observers

JavaStage has a multiple method version feature. It is possible to declare several versions of a method using multiple definitions of the configurable name. Methods with the same structure are defined once. Using these features we can develop the role and class that are depicted in Figure 4.

III. REASONS TO REFACTOR TO ROLES

In this section we present the advantages of refactoring a system to roles.

1) Refactor to Reuse Code. Delegation and inheritance may be used to reuse code. A class represents a concept others reuse by using instances of the class. If some classes have a common behavior we put that behavior in a class and make those classes inherit from it. However, with single inheritance, classes that are part of another hierarchy cannot reuse the common behavior. Multiple inheritance has many problems so many recent languages do not support it.

If we place the common behavior in a role we can reuse that role whenever we need, since they have not the multiple inheritance problems neither have single inheritance limitations. A class can play many roles and even play the same role more than once without duplicated field conflicts. The fact that roles are tailorable for a particular task, due to method renaming and type configuration allows a wider range of reuse not available with inheritance or delegation. The GenericSubject role shows how reusable a role can be.

2) Refactor to Remove Code Clones. Programmers sometimes reuse solutions by copying code and modifying it to fit a new purpose. This leads to code cloning as several fragments of a system will be identical or very similar. This can have immediate advantages like reduced development time, but in the long run a system with code clones is more difficult to maintain [15][16] and more error prone [16].

Code clones can be eliminated by better design [17] or refactoring [11][18][19]. Traditional refactoring used to deal with clones are**:** Extract Method, Pull Up Method, Extract Superclass, Extract Class and Form Template Method. We extend these refactorings by proposing to refactor to roles.

To eliminate duplicated code using roles we need to develop a role providing the replicated behavior. This way a

```
role Subject<ObserverType> {
  requires ObserverType implements
           void  #Event.update#( );
  Vector<ObserverType> observers =
           new Vector<ObserverType>();
  public void add#Observer#(ObserverType o){
     observers.add( o ); }
  void remove#Observer#(ObserverType o) {
     observers.remove( o ); }
  protected void fire#Fire#( ){
     for( FigureObserver o : observers )
        o.#Fire.update#(  );
  }
}
public class AbstractFigure implements Figure {
 plays Subject<FigureObserver>
 ( Fire=FigureMoved, Fire.update=figureMoved,
   Fire = FigurePropertyChanged,
   Fire.update = figurePropertyChanged
   Observer = FigureObserver ) figSubject;
 private Color color;
 public void moveBy(int dx, int dy) {
    fireFigureMoved( );
 }
 public void setColor( Color c ){
    fireFigurePropertyChanged( );
 }
}
```

Figure 4 A subject role and an AbstractFigure class playing it.

class does not need to replicate the code, just play the role.

3) Refactor to Enhance Modularization. A single decomposition strategy cannot adequately capture all the system's details [1]. The result are crosscutting concerns, that appear when several modules deal with the same problem because one cannot find a single module responsible for it. This leads to replicated code as each class must implement the code on its own.

With roles however, we can place the crosscutting concern in a role. The concern is thus neatly modeled. Because there is a role-player interface they can be seen as independent modules. Roles are used to compose classes but they are also independent of the classes so we can argue that roles provide a better modularization.

4) Refactor to Ease Maintenance. If a module deals with a problem that is spread by several others then changes to the code will, probably, affect other modules. Independent development is compromised. Evolution and maintenance are a nightmare because changes to that code needs to be done in all modules. If a role is used to model that concern then all changes are made in the role alone.

IV. ROLE REFACTORINGS

This section presents the role refactorings we propose. We present in tables 1 and 2, for each refactoring, the name, a summary of the situation in which the refactoring is useful and a summary of the recommended actions.

We grouped the refactorings in two categories: refactorings to extract concerns into roles (shown in table 1) and refactorings to improve role reuse (shown in table 2). We recommend that the role extraction refactorings should be used first. After the role is in place it is easier to find how we can refactor it to make it more reusable. We can also detect that some roles are similar and refactoring them so they become identical and we can leave just one.

We will present, for each refactoring, a motivation and a discussion of the mechanics. Due to space constraints we cannot present the full details but will cover the main problems and variations. We do not state where to compile and test and rely that readers are aware that these steps are crucial in refactoring. Also due to space constraints we will not present step by step snippets of code or even code samples for each refactoring but will present examples that show how several refactorings are used.

A. Refactorings to extract concerns to roles

These refactorings are intended to extract concerns to roles so classes can deal with their main concern only. There are top level refactorings like Extract Role and low level ones as Move Method Between Class and Role.

1) Extract Role. We use this refactoring whenever we feel that a class is doing work that falls outside the class main concern. The motivation is thus the same as for the Extract Class from [11].

The mechanics are simple: Create a role with a name that indicates the concern it deals with; Move each field and method that are related to that concern to the role by using Move Field From Class to Role and Move Method From Class to Role; Make the class play the role.

a) Extract Role vs Extract Class. Extract Class can be replaced by Extract Roles. This way classes do not need to create delegation methods, just play the role. Which one to use depends on the code nature. If it is a standalone concept it should be put into a class, otherwise it should be put into a role. This follows the role definition that a role is an observable behavioral aspect of a class. In Figure 1 the code reflects only a partial behavior, an entity that maintains an observer list and informs them, so role use is better.

b) Extract Role vs Extract Superclass. This refactoring could be used instead of Extract Superclass. Again the decision is based on the concept the code represents. If it is better modeled by a class and inheritance is adequate then Extract Superclass should be used. If the concept is better modeled by a role then Extract Role should be used. Extract Superclass forces classes to be in an inheritance hierarchy. In contrast, Extract Role does not require player classes to be related. On the other hand, Extract Superclass can take advantage of polymorphic code and roles cannot.

*2) Move Method from Class to Role***.** Moving a method to a role is different than moving a method to a class, so we included this refactoring. When a class plays a role it obtains the role methods, thus we do not need delegate methods. Figure 2 shows the outcome of this refactoring for the add, remove and fire methods.

The simplified mechanics are: Apply Move Method to the method always removing the delegate method; If the method makes references to the player object replace that object with the performer keyword; For each method that is called on the player place it in the requirements list.

*3) Move Field Between Class and Role***.** As with moving methods, moving a field to a role is somewhat different from moving it between classes so we decided to include a new refactoring. The main difference is that a role cannot access the player fields nor the class can access role fields.

The simplified mechanics are: If the field is used by the class from which it is being moved then use Encapsulate Field; Use Move Field on the field and Move Method on the getters and setters. Figure 2 shows the outcome of this refactoring for the vector of observers.

4) Replace Superclass with Role. Inheritance is a good way to get a default implementation for a concern. But this cannot be used just for code reuse, the classes must have something in common than just code. The benefits of reusing implementations, however, are so great that inheritance is used just the same. Roles provide another way of reusing implementations and can be used in this situation. Figure 5 shows such an example.

For this refactoring the simplified mechanics involve: Use Make Class a Role on the superclass; Replace every extends for the superclass with a plays for the role.

5) Make Class a Role. When a class only provides behavior meant to be used by other classes then it is not a class but a role (see examples in Figure 6). We use Make Class a Role by: creating a new role; Copy the code of the class into the role; If the code has references to the client type then make them refer to the Performer type; If the code

Table 1. SUMMARY OF REFACTORINGS TO EXTRACT CONCERNS TO ROLES

Refactoring name	Situation summary	Typical Action Summary
Extract Role	You have a class doing work outside its main concern	Create a role and move the relevant fields and methods to the new role
Move Method from Class to Role	A method is used or using more features from a role than the class on which it is defined	Create a new method with a similar body in the role and remove it from the class
Move Field Between Class and Role	A field is used by a role or class more than it is used by the class or role on which it is defined	Create a new field in the role or class, encapsulate it and change the class or role to access the field trough methods
Replace Superclass with Role	A superclass is used by its subclasses for reuse purposes only	Create a new role with similar code of the class and make subclasses play the role instead of inheriting from the class
Make Class a Role	You have a class that represents only a partial behavior	Make the class a role
Replace Delegation with Role Playing	A class has a number of delegating methods to another class	Create a role with similar code of the delegated class. Make the delegating class play the role instead.
Inline role	You have a role that only one class plays	Move the role code into the class
Move Method from Role to Class	A method is used or using more features from a class than the role on which it is defined	Create a new method with a similar body in the class and remove it from the role
Replace Role Playing with Superclass	Classes that are related trough inheritance are using role playing instead	Create a class that plays the role and make subclasses inherit from the class.

Table 2. SUMMARY OF THE REFACTORINGS TO IMPROVE ROLE REUSE

Refactoring name	Situation summary	Typical Action Summary
Replace Type with Generic	A role is bound to a type but could be used with another type as well	Turn the type into a generic and instantiate the type when playing the role
Make Method Name Configurable	A method name is too general to be of use in several instances of a role	Use the renaming scheme to provide a configurable name and let players configure its name.
Rename Role Method	The name of a role method does not reveal its purpose	Change the name of the method
Name a Configurable Method	A method name is configurable when it should be fixed	Remove the configurable part of the name and give the method a suitable name
Replace Generic with Type	A generic type is used in the role but players always use the same concrete type	Replace the generic type with the concrete type

has references to the client object then substitute those references to performer. For every client method referred to in the role add it to the requirements list.

6) Replace Delegation with Role Playing. A class may be used by others just to provide an implementation for some features, where the client class just delegates the job. We can have the same effect by placing the implementation in a role and the client playing the role (see Figure 6). The mechanics for this are: If the class is used in this way by all

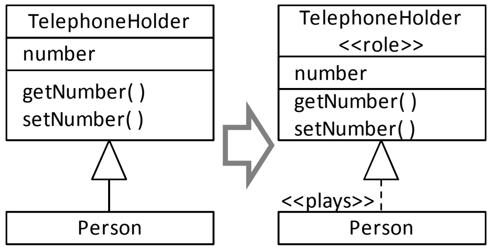


Figure 5 Replace Superclass with Role

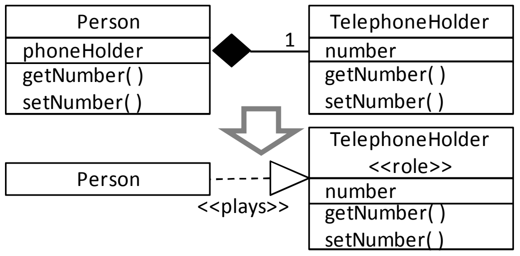


Figure 6 Replace Delegation with Role Playing

clients then use Make Class a Role; If the delegated class is used in another way by other clients consider using Extract Role on the delegated class to extract its behavior into a role; Make the class play the new role; Remove all references to the class; Remove all delegate methods.

7) Inline role. A class that plays a role has become a more suitable implementation as its concern has evolved to include that of the role, or the role is played by just one class and has an insignificant amount of behavior.

We can Inline Role by: Copying every field and method from the role to the class; If a role field has the same name of a class field Rename one so that there is not a name clash; If a class method has the same signature of a role method then do not copy that method from the role, except if the class explicitly calls that method, in which case you must Rename the role method so there is not a name clash; Delete the plays clause; Delete the role.

8) Move Method from Role to Class. This is different from Move Method From Class to Role, because of the steps involved: If the method is configurable them use the refactoring Name a Configurable Method first; If the method uses generics in the role but not on the class apply Replace Generic with Type; Apply Move Method to the method; If the method makes references to role fields use accessor methods. If the method calls other role methods make the calls explicit by using the role identity.

9) Replace Role Playing with Superclass. Classes that play the same role may be related by inheritance instead. The mechanics are: Verify if classes that use the same role using the same configurations should be related by inheritance; Create a new Class with a suitable name; Make the new class play the role using the same configurations as it subclasses; Make all classes extend the new class; Remove the plays in all subclasses.

B. Refactorings to Improve Role Reuse

We can make a role more reusable if we can expand its possible players, whether by making the types it uses more general or by making its methods configurable by the player.

1) Replace Type with Generic. Generics can be used as a place holder for the real type. The real type is defined when the code is actually being used. We suggest that if a type used by a role can be replaced by a generic it should.

Problems arise when we intend to call methods on those generic types. Java can bound a generic to certain types. For example, class Sample<T extends SuperType>, bounds T to be a subclass of SuperType. The problem when using roles is that these boundaries can be restricting. For example, in an Observer subject observers can be of any type and their interfaces are different. Roles have a requirements list that takes care of this problem.

We recommend the use of generic types instead of a concrete type. This is how to do it: Identify which types can be made generic so the role may be more reusable; Substitute each type by a generic; For each method that is called on the generic type place it in the requirements list; If the method name is not general use Rename Role Method or consider using Make Method Name Configurable.

An example is presented in Figure 3, where the FigureSubject role of Figure 2 has its FigureObserver type replaced by the generic ObserverType.

2) Make Method Name Configurable. Meaningful method names can be difficult to achieve. Role developers do not know the concrete context where the role will be used so use names that are generic. The player developer knows which names would fit the concrete use but cannot rename them because it could break other players.

To Make a Method Name Configurable: Identify which part of the method is more likely to change; Consider other methods that may have similar name parts so they can all be altered with this refactoring; Give a suitable configurable part for each method; If a method is used by the role then rename it in every place it is called and in the requirements list; The role name may also be Renamed to accommodate its wider use; Configure each player of the role so that they give the configurable methods the same names as before.

An example is presented in Figure 4. The FigureSubject role from Figure 2 is renamed to Subject and its methods are made configurable so players can choose a proper name for the add and remove methods and for each fire method.

3) Rename Role Method. Renaming a role method is trickier than renaming a class method, because the name may be configurable. If a name is not configurable then the mechanics of renaming the method is equal to Rename Method, with the difference that we must check every client of every player class. When the method is configurable the renaming is done thus: If the renaming affects only the configurable part then change it in all the role code that uses the method; Change the configurable part in all the plays clauses for that role. If the renaming affects the fixed part of the name replace it in every occurrence in the role. For each player class check if the renamed method is overridden and if it is decide if the class should not rename its own method; Change each client of each player to use the new name.

4) Name a Configurable Method. Configurable role methods allow the method name to be adequate in several situations, but sometimes we can make names configurable where a single name is suitable for every use. To Name a Configurable Method: Check if all players use the same name for the method or the name suits all players; Check if any player uses a multiple version of this method; In the role rename the method to the fixed name; In the role update all references to the method with the new name; In each player delete the configuration of the method from the plays clause if this was the only method to use that configurable part.

5) Replace Generic with Type. When developing generic roles we know we overdue the use of generics if all clients use the same concrete type. This refactoring makes the role simpler to use. The mechanics are: Replace all occurrences of the generic with the concrete type in the role; If the generic has entries in the requirement list delete them; In each player remove the instantiation of the concrete type.

V. RELATED WORK

There is much work related to Object Oriented refactorings [11][12] and we adapted some of those to roles, but to our knowledge there is no published work that concerns refactoring to roles. This includes the works of dynamic roles and not just static roles. Static roles have been used in the work of VanHilst and Notkin in [20] where they proposed to use roles in the C++ language. Dynamic role approaches as EpsilonJ [21] and PowerJava [10] have been around for a while but no refactorings to dynamic roles have been published. We believe that our adaptation of code smells to roles can also benefit these role related approaches.

Object Teams in its project home page [22] mentions the adaptation of Extract Method, Move Method, Pull Up, Pull Down and Rename to the objects teams specific relationships (implicit role inheritance, team nesting, role-base bindings and method bindings). They also support new role related refactorings like Extract Callin and Inline Callin. But, there is not a presentation or mechanics of these refactorings.

The role object pattern [23] is used for representing objects that expose different properties in different contexts. Steimann and Stolz [24] describe a way to refactor code to this pattern that provides lightweight role objects with a leaner code than the previous approaches. They also softened the preconditions on when to apply the refactoring.

There are other approaches to class compositions, like Traits [3], Multi-dimensional separation of concerns [1]. Package Templates (PT) [25], Caesar and its Virtual classes [26], Jiazzi and its Units [27]. To our knowledge none of these approaches tackled the problem of refactoring legacy code. We consider Traits to be the most related approach to static roles, as we can see a trait as a role without state. We believe, therefore, that some role refactorings can be used in Traits, namely Extract Role could be used as an Extract Trait as long as we removed the part related to moving fields and replaced it with Encapsulate Field.

Feature Oriented Programming (FOP) decomposes the system into features [3]. Features reflect user requirements and incrementally refine each other. In [28], Liu et al propose a theory of Feature Oriented Refactoring (FOR), which is the process of decomposing a program into features, thus recovering a feature based design and giving it an important form of extensibility. Since a feature's implementation can vary between systems, the authors developed an algebraic theory of FOR that exposes the highly regular structure that features impose on programs. They also supply a methodology and a tool based on the theory. This work, however, can be applied only to FOP.

Aspect-Oriented Programming as used in AspectJ [5] is an approach that tries to modularize crosscutting concerns. There is work on refactorings systems to aspects [29]. Due to the renaming capability of JavaStage we can include some refactorings related to method names, while in AOP we cannot.

VI. Conclusions and Future work

We showed that refactoring a system to roles brings benefits to the system like a higher reusability, better modularization, among others.

We proposed a series of refactorings based on our studies with converting OO systems to roles and design pattern implementation using roles. These refactorings provide a way to convert legacy code to role code. Some refactorings deal with the problem of making the role more general purpose thus enhancing code reuse.

For future work we intend to develop a tool to give these refactorings some automatic support. We also intend to carry on our studies concerning role development so we can discover new refactorings that involve role development and use, not just upgrading roles and refactoring to roles. This will contribute to a more complete role refactoring catalogue.

References

[1] P. Tarr, H. Ossher, W. Harrison, and S. M. Sutton Jr., "N degrees of separation: multi-dimensional separation of concerns", Proc. International Conference on Software Engineering, 1999, pp. 107-119.

[2] G. Bracha and W. Cook, "Mixin-based inheritance". Proc. OOPSLA/ Proc. ECOOP, 1990, pp 303-311.

[3] S. Ducasse, N. Schaerli, O. Nierstrasz, R. Wuyts and A. Black, "Traits: a mechanism for fine-grained reuse". Transactions on Programming Languages and Systems. vol. 28, no. 2, March 2006, pp. 331-388.

[4] S. Apel and C. Kästner, "An overview of Feature-Oriented Software Development", in Journal of Object Technology, vol. 8, no. 5, July–August 2009, pp 49–84.

[5] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W.G. Griswold, "An overview of AspectJ". Proc. ECOOP 2001, pp 327-353.

[6] D. Riehle and T. Gross, "Role model based framework design and integration", Proc. OOPSLA '98. 1998, pp. 117-133.

[7] F. Steimann, "On the representation of roles in object-oriented and conceptual modeling", Data & Knowledge Engineering, vol. 35, no. 1, 2000, pp 83–106.

[8] F. Barbosa and A. Aguiar, (2013), "Using roles to model crosscutting concerns", Proc. Aspect Oriented Software Development (AOSD13), March 2013, pp 97-108.

[9] S. Herrmann, "Programming with Roles in ObjectTeams/ Java". AAAI Fall Symposium: Roles, An Interdisciplinary Perspective, 2005.

[10] M. Baldoni, G. Boella and L. van der Torre, "Interaction between objects in power-Java", Journal of Object Technologies, vol 6, 2007, pp 7 – 12.

[11] M. Fowler. "Refactoring – Improving the Design of Existing Code", Addison Wesley, 2000.

[12] W. Opdyke, "Refactoring Object-Oriented frameworks", Ph.D. thesis, Univ. of Illinois at Urbana-Champaign, 1992.

[13] F. Barbosa and A. Aguiar, "Generic roles, a test with patterns" Proc. Pattern Languages of Programs, 2011.

[14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "Design Patterns: elements of reusable Object-Oriented software", Addison-Wesley, 1995.

[15] I. Baxter, A. Yahin, L. Moura, M. Sant'Anna, and L. Bier, "Clone detection using abstract syntax trees", Proc. of the Int. Conf. on Software Maintenance, Nov. 1998, pp 368-377

[16] E. Juergens, F. Deissenboeck, B. Hummel, and S. Wagner, "Do code clones matter?", Proc. Int. Conf. on Software Engineering, IEEE Computer Society, 2009, pp 485-495

[17] M. Balazinska, E. Merlo, M. Dagenais, B. Lague, and K. Kontogiannis, "Measuring clone based reengineering opportunities". Proc. International Software Metrics Symposium, Nov. 1999, pp 292-303

[18] R. Fanta and V. Rajlich, "Removing clones from the code". Journal of Software Maintenance: Research and Practice, vol. 11, no 44, August 1999, pp 223-243.

[19] Y. Higo, T. Kamiya, S. Kusumoto, and K. Inoue. "Refactoring support based on code clone analysis". Proc. International Conference on Product Focused Software Process Improvement, 2004, pp 220-233.

[20] M. VanHilst and D. Notkin, (1996) "Using role components to implement collaboration-based designs". Proc. OOPSLA '96, 1996, pp 359-369.

[21] T. Tamai, N. Ubayashi, and R. Ichiyama, "Objects as actors assuming roles in the environment", in Software Engineering For Multi-Agent Systems V: Research Issues and Practical Applications, Lecture Notes In Computer Science, vol. 4408. Springer-Verlag, 2007, pp 185-203

[22] http://www.eclipse.org/objectteams/features.php, last access in Jan. 2013.

[23] D. Bäumer, D. Riehle, W. Siberski, and M. Wulf, "The role object pattern", Proc. PLoP1997, 1997. pp 15-31

[24] F. Steimann and F. U. Stolz,. "Refactoring to role objects", Proc. International Conference on Software Engineering, 2011, pp 441-450.

[25] S. Krogdahl, B. Møller-Pedersen, and F. Sørensen, "Exploring the use of Package Templates for flexible re-use of Collections of related Classes", Journal of Object Technology, vol. 8, no. 7, Nov. – Dec. 2005, pp 59-85.

[26] E. Ernst, K. Ostermann, and W. R. Cook. "A virtual class calculus", Conference record of the 33rd Symposium on Principles of Programming Languages. 2006, pp 309-330.

[27] S. McDirmid, M. Flatt, and W.C. Hsieh, "Jiazzi: new-age components for old-fashioned Java", Proc. OOPSLA 2001, pp 211-222.

[28] J. Liu, D. Batory, and C. Lengauer, "Feature oriented refactoring of legacy applications". Proc. Inter. Conference on Software Engineering (ICSE '06), 2006, pp 872-881.

[29] M. Monteiro and J. Fernandes. "Towards a catalog of aspect-oriented refactorings". Proc. Int. Conf. on Aspect-Oriented Software Development, 2005, pp 111–122.

Linking E-Mails and Source Code Using BM25F

Raffaele Branda, Anna Tolve, Licio Mazzeo, and Giuseppe Scanniello
University of Basilicata, Potenza, ITALY
Email: raf.bran@gmail.com, anna.tolve@tiscali.it, licio.mazzeo@gmail.com, giuseppe.scanniello@unibas.it

***Abstract*—Existing approaches to recover links between e-mails and software artifacts are based on text search or text retrieval and reformulate link recovery as a document retrieval problem. We refine and improve such solutions by leveraging the parts of which an e-mail is composed of: header, current message, and previous messages. The relevance of these parts is weighted by a probabilistic approach based on text retrieval. We implemented our novel solution exploiting the BM25F model. The results of an empirical study conducted on a public benchmark indicate that the new approach in many cases outperforms the baseline approaches chosen. In addition, the proposed approach is easy to use and it is accurate enough to be worth the costs it may introduce in the corpus preprocessing and indexing.**

Keywords - ***Empirical Study; Probabilistic Approach; Traceability Recovery***

I. Introduction

Maintenance operations are carried out for several reasons and are typically classified as corrective, perfective, and adaptive [1]. Whatever is the maintenance operation, the greater part of the cost and effort is due to the comprehension of source code [2]. Pfleeger and Atlee [3] estimated that up to 60% of software maintenance is spent on the comprehension of source code. There are several reasons that make source code comprehension even more costly and complex and range from the size of the subject software to its overall quality. Other reasons are related to the knowledge of a subject system that is implicitly expressed in software artifacts (i.e., models, documentation, source code, e-mails, and so on) [4]. This knowledge is very difficult to retrieve and it is very often enclosed in non-source artifacts [5].

Among non-source artifacts, those composed of free-form natural language (e.g., documentation, wikis, forums, e-mails) are intended to be read by stakeholders with different experience and knowledge (e.g., managers, developers, testers, and end-users). This kind of artifacts often implicitly or explicitly references to other forms of artifacts, such as source code [6]. Linking e-mails and source code could improve software comprehension and could help to understand the justification behind decisions taken during the design and development [7]. Then, links between e-mails and source code are worthwhile within the entire software lifecycle and in software maintenance, in particular (e.g., [4], [8]).

Several approaches have been proposed to recover links among software artifacts (e.g., [9], [10], [11]). Only a couple of them are concerned with e-mails [12], [13] and can be classified as: rule-based and Information Retrieval (IR) based.

Rule-based. To detect latent links between emails and source code entities hand-code specific rules (i.e., sets of regular expressions) have to be specified. These rules are in turn triggered whenever they match with a portion of email text (e.g., [6]). For example, if the identifiers in the source code repository follows the CamelCase naming convention, we basically know that each identifier is either a single or a compound name (i.e., a sequence of unseparated single names). In the case of class names, all the single names start with a capital letter. Therefore, we can define a regular expression so that every time we find a string in an e-mail of the form `Foo`, `FooBar`, `FooBarXYZ`, etc., we can mark it as a link between the source code and the e-mail. This kind of approach is computationally lightweight for small/medium corpora (e.g., repositories with a small number of e-mails) and easy to implement. Conversely, they lack of flexibility since they are strictly programming-language-dependent. Even more, they do not provide any ranking score associated with the discovered link (i.e., information about a link is binary: a link is either present or not).

IR-based. These approaches reformulate the problem as a particular instance of the more general document retrieval problem. They use IR techniques to compare a set of source artifacts (software entities) with a set of target artifacts (e-mails). Each source code entity (e.g., the class name) is used as the query to retrieve the set of most relevant e-mails. Candidate links are then devised by inspecting the ranked list of retrieved e-mails. Relevance between any pair of source and target artifacts (i.e., source code entity and email) can be determined by their textual/lexical similarity, which is computed by using a specific IR model in conjunction with a particular term-weighting score (e.g., cosine similarity using *tf-idf* vector space model) [14]. The main advantage of IR-based approaches is that they are more flexible and associate each discovered link with a ranking score.

In this paper, we propose an IR-Based approach that refines and improves existing solutions by leveraging the parts of which an e-mail is composed of, namely the header, the current message (from here on, body), and the sentences from previous messages (quote). The relevance of these parts has been weighted by means of a text retrieval probabilistic model. In particular, we implemented our novel solution exploiting the BM25F model [15], [16]. This model is based on a term weighting scheme which takes into account the fact that semi-structured documents from a corpus can be composed of fields [17]. These fields differently contribute to the representation of documents and then to the accuracy of the links retrieved. To assess the validity of our proposal, we have conducted an empirical study on the public benchmark proposed by Bacchelli *et al.* [12].

Structure of the paper. We illustrate our approach in Section II. In Section III, we present the design of the empirical evaluation, while we discuss the achieved results in Section IV. Final remarks conclude the paper.

II. The Approach

IR-based traceability recovery approaches reformulate traceability recovery as a document retrieval problem. We refine and improve such solutions by leveraging header, body, and quote of e-mails. We describe the steps of our approach in the following subsections.

A. Creating a Corpus

Each e-mail results in one document in the corpus. Each document has three well defined fields: header, body, and quote. The header field contains the subject of an e-mail, while the body the sentences of the current message. All the sentences from previous messages are within the quote field. In particular, it includes a chain of messages (e.g., ideas, opinions, issues, or possible solutions) exchanged among stakeholders (mostly developers) linked in the sequence in which they espoused that discussion. We also consider the quote because IR approaches produce better results when a huge amount of lexical information is available [14]. Moreover, the body and the quote fields are separately considered since the lexical information within the body is on the current focus of a discussion, while the quote field includes text that might provide useful information on the entire discussion thread.

B. Corpus Normalization

The corpus is normalized: *(i)* deleting non-textual tokens (i.e., operators, special symbols, numbers, etc.), *(ii)* splitting terms composed of two or more words (e.g., `first_name` and `firstName` are both turned into `first` and `name`), and *(iii)* eliminating all the terms within a stop word list (including the keywords of the programming languages: Java, C, ActionScript, and PHP) and with a length less than three characters. We applied these normalization rules because they have been widely applied in IR-based traceability recovery approaches (e.g., [11]).

Splitting identifiers could produce some noises in the corpus. For example, if the name of a class is `FileBuffer`, it is possible that a software engineer talks about FileBuffer in an e-mail rather than `File` and `Buffer`. However, if the identifiers are not split the things could go from bad to worse: the class name is not in the text of the e-mails (e.g., [12] and [13]), while that name is used as the query. To deal with this issue, we apply the same normalization process on both the corpus and the queries and use the "AND" operator to formulate each query.

Differently from the greater part of the traceability recovery approaches (e.g., [9], [11]), we did not apply any stemming technique [14] to reduce words to their root forms (e.g., the words `designing` and `designer` have `design` as the common radix). This is because we experimentally observed that the use of a Porter stemmer [18] led to worse results. Also, in [12] the stemming was not used for similar reasons.

C. Corpus Indexing

We adopt here a probabilistic IR-based model, namely BM25F [15]. This model extends BM25 [16] to handle semistructured documents from a corpus. The BM25 model was originally devised to pay attention to term frequency and document length, while not introducing a huge number of parameters to set [19]. BM25 showed very good performances [16] and then widely used specially in web document retrieval applications [17], [20]. BM25F was successively proposed to build a term weighting scheme considering the fact that documents from a corpus can be composed of fields (e.g., [17]). Each document is in the corpus and contains information on the contained fields. Then, the fields of a document differently contribute to the document representations. We used BM25F because it has been successfully used on very large corpuses [20] in terms of both scalability and quality of retrieved documents [21]. The use of other probabilistic models could lead to different results. This point is subject of future work.

In both "vector space" and "probabilistic" IR methods, an information retrieval scheme is built for considering each document as a point in a multi-dimensional geometrical space. Therefore, BM25F is based on the bag-of-words model, where each document in the corpus is considered as a collection of words disregarding all information about their order, morphology, or syntactic structure. A word could appear in different fields of the same document. In this case, that word is differently considered according to the field in which it appears. Applying BM25F, each e-mail in the corpus is represented by an array of real numbers, where each element is associated to an item in a dictionary of terms. BM25F does not use a predefined vocabulary or grammar, so it can be easily applied to any kind of corpora.

BM25F works on the occurrence of each term in the fields of all the documents in the corpus. These occurrences are used to build a *term-by-document* matrix. In the current instantiation of this step we modified the original definition of BM25F to better handle the problem at hand. In the model, a generic entry of the table is computed as follows:

$$idf(t,d) = \log\left(\frac{N - df(t) + 0.5}{df(t) + 0.5} + 1\right) * weight(t,d) \quad (1)$$

where N is the total number of documents in the corpus, while df is the number of documents where the term t appears. The weight of the term t with respect to the document d is computed by $weight(t,d)$ as follows:

$$weight(t,d) = \sum_{c \ in \ d} \frac{occurs^{d}_{t,c} * boost_c}{((1-b_c) + b_c * \frac{l_c}{avl_c})} \quad (2)$$

l_c is the length of the field c in the document d; avl_c is the average length of the field c in all the documents; and b_c is a constant related to the field length; and $boost_c$ is the boost factor applied to the field c. $occurs^{d}_{t,c}$ is the number of terms t that occur in the field c of the document d. This equation is dependent on the field and document relevance and it is similar to a mapping probability. This is because BM25F is considered a probabilistic IR-based model. Regarding the constants of (1), we chose 0.75 as the value for b_c, while 1 is the boost value applied to each field (i.e., header, body, and quote). These values were experimentally chosen and are customary in the IR field [21].

In the original definition of BM25F [20], if a term occurs in over half the documents in the corpus, the model gives a negative weight to the term. This undesirable phenomena is well established in the literature [14]. It is rare in some applicative contexts, while it is common in others as for an example in the recovery of links between e-mails and source code. In such a context, in fact, e-mails quote sentences from previous messages and then the difference among e-mails (in the same discussion thread) is not that great with respect to the terms contained. To deal with this concern, we modified the computation of idf. The adopted solution is that shown in the equation (1), which is based on that suggested in [22]. The main difference with respect to the canonical computation of idf is that 1 is added to the argument of the logarithm.

D. Query Formulation

In the traceability recovery field, source artifacts are used as the query [9]. The number of queries is then equal to the number of source artifacts. In this work, we used source code entities as the source artifacts and applied the following two instantiations for Query Formulation: *(i)* class names and *(ii)* class and package names. We opted here for that solution because we wanted to compare our novel approach with some baselines (included those in [12]) on a public benchmark [13]. In addition, this solution allowed us to automatically formulate "semi-structured" queries directly parsing the source code[1].

The queries are normalized in the same way as the corpus. When the textual query is composed of more than one term (e.g., `ArgoStatusBar`), the boolean operator "AND" is used with the individual terms of that query (`Argo`, `Status`, and `Bar`). This implies that all the individual terms have also to exist anywhere in the text of a document.

E. Ranking Documents

For a probabilistic IR method, the similarity score between a query with a document in the corpus is not computed by the cosine similarity, but by a different formula motivated by the probability theory [14]. In this work, we used a formula based on a non-linear saturation to reduce the effect of term frequency. This means that the term weights do not grow linearly with term frequency but rather are saturated after a few occurrences:

$$score(q,d) = \sum_{t \ in \ q} idf(t) * \frac{weight(t,d)}{k_1 + weight(t,d)} \qquad (3)$$

where q is the textual query and d is a document in the corpus. The values for $idf(t)$ and $weight(t,d)$ are computed as shown in the equations (1) and (2), respectively. The parameter k_1 usually assumes values in the interval $[1.2, 2]$. We used 2 as the value because experiments suggested that it is a reasonable value [14] to maximize retrieval performances.

F. Examining Results

A set of source artifacts is compared with set of target artifacts (even overlapping). Then, all the possible pairs (candidate links) are reported in a ranked list (sorted in descending order). The software engineer investigates the ranked list of candidate links to classify them as actual or false links.

[1]Different kinds of queries can be formulated automatically or not. For example, the source code content could be also used as the query. We experimentally observed that the use of this kind of query leads to worse results with respect to the other two kinds of query we prose here. Therefore, we did not consider this instantiation for the Query Formulation step. This result is in line with that of Bacchelli *et al.* [12].

III. Empirical Evaluation

Based on the above instantiation of our approach, we implemented an Eclipse plug-in, named Linking e-mAils and Source COde (LASCO). This plug-in has been described in a previous tool demo paper [23]. To asses both the approach and the plug-in, we conducted an empirical study (i.e., an experiment). The presentation of that study is based on the guideline suggested in [24].

A. Definition

As suggested in [24], the goal of our study has been defined using the Goal Question Metrics (GQM) template [25]: ***Analyze*** *traceability recovery links between e-mails and source code* ***for the purpose of*** *evaluating the use of BM25F on header, body, and quote* ***with respect to*** *the accuracy of the retrieved links* ***from the point of view of*** *the researcher and the practitioner* ***in the context of*** *open source systems.*

To this end, we then formulated and investigated the following research question: ***Does our proposal outperform baseline approaches based on text search or text retrieval methods?*** We considered in this study the following baselines:

1. BM25F with the "OR" operator: We apply the BM25F model and the "OR" operator in the step Query Formulation. The Corpus Indexing step is executed by considering the e-mails as composed of header, body, and quote. The only difference with respect to our proposal is that the "OR" operator is used against the "AND" operator;
2. BM25F considering body and quote together: We apply the BM25F model and the operators "AND" and "OR". Furthermore, the Corpus Indexing step is performed considered two fields: *(i)* header and *(ii)* body and quote together;
3. Lucene with "AND" and "OR" operators: In the Corpus Indexing step, we use Lucene. It uses a combination of Vector Space Model (VSM) and the Boolean model to determine how relevant a document is to a query. We here apply both the operators "AND" and "OR". Since Lucene is based on VSM, more times a query term appears in a document relative to the number of times the term appears in all the documents in the corpus, the more relevant that document to the query is;
4. VSM: It represents the documents in the corpus as term vectors, whose size is the number of terms present in the vocabulary. Term vectors are aggregated and transposed to form a *term-document matrix*. To take into account the relevance of terms in each document and in all the corpus, many weighting schema are available. In our empirical evaluation, we employed the *tf-idf* (term frequency - inverse document frequency) weighting;
5. LSI: Even for a corpus of modest size, the term-document matrix is likely to have several tens of thousand of rows and columns, and a rank in the tens of thousands as well. LSI is an extension of VSM developed to overcome the synonymy and polysemy problems [26]. SVD (Singular Value Decomposition) is used to construct a low-rank approximation matrix to the term-document matrix [27]. In LSI there is no way to enforce Boolean conditions [14];
6. Lightweight linking technique (LLT) - case sensitive (CS): To reference software entities from e-mails, the names of the software entities are used as text search queries. There exists a link between a software entity and an e-mail, when

there is a case sensitive match on the entity name;
7. LLT - mixed approach (MA): In case the name of software entities are compounded words, they are split (e.g., `ClassName` becomes `Class Name`). The compounded words are then used for the case sensitive match on the entity name, otherwise it is used a regular expression based on class and package name;
8. LLT - MA with regular expression (RE): This approach is based on that above. A different regular expression is used to better handle non-Java systems. Further details about Lightweight linking techniques can be found in [12].

The baselines from 1 to 5 are different instantiations of the recovery process shown in Section II, while the others are lightweight approaches based on regular expressions. In all the IR-based baseline approaches, with the exception of the first and second one, the corpus was indexed considering together header, body, and quote.

B. Planning

1) Context: Many IR-based traceability recovery approaches depend on users' choices: the software engineer analyzes a subset of the ranked list to determine whether each traceability link has been correctly retrieved. It is the software engineer who makes the decision to conclude this process. The lower the number of false traceability links retrieved, the better the approach is. The best case scenario is that all the retrieved links are correct. IR-based traceability recovery methods are far from this desirable behavior [9]. In fact, IR-based traceability recovery approaches might retrieve links between source and target artifacts that do not coincide with correct ones: some are correct and others not. To remove erroneously recovered links from the candidate ones, a subset of top links in the ranked list (i.e., retrieved links) should be presented to the software engineer. This is possible by selecting a threshold to cut the ranked list (e.g., [11], [28]).

There are methods that do not take into account the similarity between source and target artifacts: *Constant Cut Point*, it imposes a threshold on the number of recovered links, and *Variable Cut Point*, it consists in specifying the percentage of the links of the ranked list to be considered correctly retrieved. Alternative possible strategies for threshold selection are based on the similarity between source and target artifacts: *Constant Threshold*, a constant threshold is chosen, *Scale Threshold*, a threshold is computed as the percentage of the best similarity value between two vectors, and *Variable Threshold*, all the links among those candidate are retrieved links whether their similarity values are in a fixed interval. In our experiment, we used the Constant Threshold method. This is the standard method used in the literature [9]. We applied this method employing thresholds assuming values between 0 and 1. The increment used was 0.01.

For each software entity, the Query Formulation step was instantiated using either the original class name or the concatenation of class and package names. Many of the design choices have been taken because our main goal was to compare the results of our solution with those presented in [12].

2) Variable selection: The traceability links retrieved by applying both our approach and the baselines are analyzed in terms of *correctness* and *completeness*. Correctness reflects the fact that an approach is able to retrieve links that are correct. To measure the correctness, we used (as customary) *precision* ($precision = \frac{|TP|}{|TP|+|FP|}$). On the other hand, completeness reflects how much the set of retrieved links is complete with respect to the all actual links. *Recall* is used to measure this aspect ($recall = \frac{|TP|}{|TP|+|FN|}$). where TP (true positives) is the set of links correctly retrieved. The set FN (false negatives) contains the correct links not retrieved, while FP (false positives) the links incorrectly presented as correct.

When the e-mails in the benchmark do not have any reference to source code artifacts, the union of TP and FN is empty (i.e., $|TP| + |FN| = 0$). In all these cases, we cannot calculate the values for the recall measure. The values for precision could not be computed in case the approach found no link between an e-mail and the source code. Similar to [12], we avoided these issues calculating the average of $|TP|$, $|FP|$, and $|FN|$, on the entire dataset. We then computed the average values for precision and recall. Precision and recall assume values in the interval $[0,1]$. The higher the precision value, the more correct the approach is. Similarly, the higher the recall value, the better the approach is.

To get a trade-off between correctness and completeness, we applied the balanced F-measure (i.e., $F_1 = \frac{2*precision*recall}{precision+recall}$). F_1 was used to estimate the *accuracy* of the approach. This is the main criterion we considered in the study. This measure has values in the interval $[0,1]$. When comparing two approaches, the one with higher F_1 value is considered the best, namely the most accurate.

3) Instrumentation: To estimate our approach and to compare it with the baselines, we used the benchmark proposed in [13]. For each system and all the threshold values, we computed the values of precision, recall, and F_1. To compare our approach with the baselines, we selected the constant threshold that produced the best accuracy.

IV. Results and Discussion

A. Results

The results achieved by applying our approach are shown in Table I. The table also reports the results achieved by applying the "OR" operator. The results are grouped according to the two different instantiations of the step Query Formulation: *(i)* class name and *(ii)* class and package names. The last row reports the average values for each measure. Better average accuracy was achieved using class and package names and the "AND" operator (F_1 = 0.44). With respect to each individual system, we obtained the higher accuracy for Habari, namely the system implemented in PHP (F_1 = 0.59). On that system, the higher value of correctness was also obtained (precision = 0.77). It is worth mentioning that the results for that system are the same both using class name alone and class and package names together. This is because PHP 5 did not have packages. Namespaces (i.e., packages) where only introduced in PHP 5.3. The same held for Augeas (the C software system).

Table II shows the results achieved by indexing the corpus using: *(i)* header and *(ii)* body and quote together. With respect to accuracy, better results were achieved using the operator "AND" and class and package names. The best average accuracy value was 0.41. Among the analyzed software systems, the best accuracy was obtained for Habari (F_1 = 0.61).

TABLE I. BM25F RESULTS INDEXING THE CORPUS USING: *(i)* HEADER, *(ii)* BODY, AND *(iii)* QUOTE

	Class Name + "AND"			Class Name + "OR"			Class and Package Names + "AND"			Class and Package Names + "OR"		
System	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
ArgoUML	0.32	0.53	0.40	0.05	0.55	0.10	0.41	0.72	0.52	0.04	0.48	0.07
Freenet	0.23	0.49	0.31	0.03	0.23	0.06	0.30	0.52	0.39	0.02	0.40	0.05
JMeter	0.32	0.41	0.36	0.10	0.41	0.16	0.49	0.62	0.55	0.06	0.43	0.10
Away3D	0.31	0.51	0.39	0.15	0.24	0.18	0.39	0.44	0.41	0.12	0.24	0.16
Habari	0.77	0.48	0.59	0.29	0.35	0.32	0.77	0.48	0.59	0.29	0.35	0.32
Augeas	0.12	0.27	0.16	0.04	0.32	0.08	0.12	0.26	0.16	0.04	0.32	0.08
Average value	0.35	0.45	0.37	0.11	0.35	0.15	0.41	0.51	0.44	0.10	0.37	0.13

TABLE II. BM25F RESULTS INDEXING THE CORPUS USING: *(i)* HEADER AND *(ii)* BODY AND QUOTE TOGETHER

	Class Name + "AND"			Class Name + "OR"			Class and Package Names + "AND"			Class and Package Names + "OR"		
System	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
ArgoUML	0.34	0.51	0.41	0.07	0.58	0.12	0.40	0.46	0.43	0.05	0.55	0.09
Freenet	0.22	0.54	0.31	0.08	0.45	0.14	0.29	0.62	0.40	0.07	0.5	0.13
JMeter	0.29	0.45	0.36	0.14	0.41	0.21	0.34	0.66	0.45	0.12	0.45	0.19
Away3D	0.29	0.76	0.42	0.21	0.24	0.22	0.37	0.44	0.40	0.16	0.23	0.19
Habari	0.74	0.52	0.61	0.46	0.45	0.46	0.74	0.52	0.61	0.46	0.45	0.46
Augeas	0.11	0.35	0.17	0.10	0.17	0.13	0.11	0.35	0.17	0.06	0.35	0.10
Average value	0.33	0.52	0.38	0.18	0.38	0.21	0.38	0.5	0.41	0.15	0.42	0.19

TABLE III. LUCENE RESULTS

	Class Name + "AND"			Class Name + "OR"			Class and Package Names + "AND"			Class and Package Names+ "OR"		
System	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
ArgoUML	0.32	0.50	0.39	0.06	0.50	0.11	0.39	0.47	0.43	0.03	0.53	0.06
Freenet	0.20	0.59	0.30	0.07	0.47	0.11	0.27	0.64	0.38	0.05	0.56	0.10
Jmeter	0.27	0.46	0.34	0.10	0.36	0.15	0.34	0.70	0.46	0.07	0.49	0.13
Away3D	0.29	0.77	0.42	0.17	0.22	0.19	0.37	0.44	0.40	0.13	0.24	0.17
Habari	0.61	0.55	0.58	0.45	0.40	0.43	0.61	0.55	0.58	0.45	0.40	0.42
Augeas	0.10	0.27	0.15	0.05	0.21	0.08	0.10	0.27	0.15	0.05	0.20	0.08
Average value	0.30	0.52	0.36	0.15	0.36	0.18	0.35	0.51	0.40	0.13	0.40	0.16

TABLE IV. RESULTS BY BACCHELLI *et al.* [12]

	VSM with $tf-idf$			LSI			LLT - case sensitive			LLT - mixed approach			LLT - mixed approach with RE		
System	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1	*Precision*	*Recall*	F_1
ArgoUML	0.25	0.34	0.29	0.60	0.48	0.53	0.27	0.68	0.38	0.64	0.61	0.63	0.35	0.68	0.46
Freenet	0.15	0.25	0.19	0.62	0.43	0.51	0.17	0.70	0.27	0.59	0.59	0.59	0.27	0.69	0.39
JMeter	0.21	0.34	0.26	0.52	0.40	0.45	0.15	0.73	0.25	0.59	0.65	0.62	0.30	0.72	0.42
Away3D	0.35	0.31	0.33	0.35	0.33	0.34	0.32	0.74	0.44	0.40	0.54	0.46	0.41	0.72	0.52
Habari	0.34	0.39	0.36	0.36	0.41	0.38	0.40	0.41	0.41	0.83	0.09	0.17	0.49	0.38	0.43
Augeas	0.10	0.20	0.14	0.10	0.28	0.14	0.09	0.72	0.15	0.14	0.02	0.04	0.15	0.64	0.24
Average value	0.23	0.31	0.26	0.43	0.39	0.39	0.23	0.66	0.32	0.53	0.42	0.42	0.33	0.64	0.41

The results achieved with Lucene are shown in Table III. The best average accuracy value was reached using the operator "AND" and class and package names (i.e., 0.40). The better accuracy was achieved for Habari (F_1 = 0.58).

Table IV summarizes the results presented in [12], instantiating Query Formulation step with class name. As mentioned before, the results for class and package names together are not reported for VSM and LSI because the authors observed that better results were achieved using only class names. Table IV also shows the results for the lightweight linking techniques.

The results indicate that our proposed technique is more accurate than BM25F using two fields (header and body and quote together) on all the Java systems with the exception of Freenet (the F_1 values were 0.39 and 0.40, respectively). On the non-Java systems, the use of BM25F indexing the corpus with three or two fields did not produce remarkable differences in accuracy (see Table I and Table II).

Our approach using class and package names as the queries is more accurate than VSM. Similar results were achieved for Lucene using both the operators and class name and class and package names together as the queries. Indeed, our proposal did not outperform Lucene only on Away3D when using the "AND" operator and class name as the query. The F_1 values were 0.41 and 0.42, respectively.

As far as LSI, our approach is more accurate on all the non-Java system and Jmeter. For ArgoUML the difference in favor of LSI was negligible (the F_1 values was 0.52 with respect to 0.53). A larger difference in accuracy was obtained for Freenet.

Our proposal outperformed LLT-CS in accuracy on all the systems with the exception of Away3D (the F_1 values were 0.44 and 0.41, respectively). BM25F with three fields was on average more accurate than LLT MA with and without RE (see the average values of F_1). With respect to LLT MA, we achieved better F_1 values results on Habari and Augeas (0.59 vs. 0.17 and 0.16 vs. 0.04, respectively). On the Java systems LLT MA was more accurate than our approach. For LLT MA RE, we reached better results on the Java systems and Habari.

Regarding the correctness and completeness of the retrieved links, we can observe an interesting pattern in the data: our approach mostly allowed obtaining a more complete set of retrieved links that are correct. This result is desirable when you are interested in the recovery of links among software artifacts (e.g., [9]).

B. Discussion

1) IR-Based recovery: For Java systems, LSI outperformed other approaches based on IR techniques with respect to the accuracy of the retrieved links. A reason is that each e-mail in the corpus quotes a large number of sentences from previous messages. This is the best scenario for using LSI [14]. In fact, this technique is used to disclose the latent semantic structure of a corpus and to recognize its topics, so dealing with synonymy and polysemy problems. Further, each document in the corpus has a large size as compared with the entities used as the queries. This might also represent another possible reason for having achieved better results on Java systems. The considerations above and the fact that LSI outperformed our approach in terms of accuracy only on Freenet (this difference was 0.04, while this difference was in favor of our approach on ArgoUML and JMeter and was 0.01 and 0.1, respectively) suggest that BM25F represents an alternative also when dealing with large documents in the corpus.

In the case that e-mails in the corpus quote a small number of sentences from previous e-mails our approach outperformed other baseline approaches based on IR techniques. This happened for all the non-Java systems. For the Habari system, the e-mails were very short and then BM25F made the difference also considering the information in the body and quote together.

For the system implemented in C (i.e., Augeas), the application of the IR-Based approaches mostly produced worse results in terms of correctness, completeness, and accuracy. As also suggested in [12], a possible justification is related to the names of the entities. However, our approach outperformed the IR based baselines. Again, indexing the e-mails considering two or three fields did not produce remarkable differences.

The instantiation of the Query Formulation step with class and package names improved the correctness and completeness when our technique was used. Then, it is possible that the choice of the source artifact can make the difference in the accuracy of the links recovered.

The use of a stemming technique in the Normalization step produced worse results. Then, this technique seems useless in the recovery of links between source code and e-mails, when using BM25F (with two and three fields) and Lucene. On these instances, the use of the "AND" operator led to better results in terms of accuracy and correctness of the retrieved links with respect to the "OR" operator. This result held for all the systems. For completeness, the results achieved with the "AND" operator were mostly better than those achieved with the "OR" operator. Only in four cases the use of the "OR" operator led to better recall values.

The use of source code (program statements and/or source code comment) as the query was also analyzed. The results revealed that this kind of instantiation for the Query Formulation step led to worse results with respect to the other two kinds of queries considered here. This result is in line with that shown in[12] and has the following implication: it is better to use class name and class and package names as the queries.

We also performed an analysis to get indications on whether BM25F might introduce scalability issues. We used a laptop equipped by a processor Intel Core i7-2630QM with 4 GB of RAM and Windows Seven Home Premium SP-1 64bit as operating system. This analysis was performed on each system and the baseline processes implemented for our experiment (see Section III-A). The results indicated that the time to build, normalize, and index the e-mails of the entire benchmark was twice when using three fields (i.e., 5033 milliseconds) with respect to the use of two fields (i.e., 2668 milliseconds). For Lucene, the average execution time on all the systems in the benchmark was 2660 milliseconds. For the Query Formulation step, nearly the same pattern was observed. Further details are not provided for space reason.

2) Lightweight Approaches: Regarding the accuracy of the retrieved links, LLT MA outperformed the other lightweight techniques and our approach on the Java systems. On the non-Java system with the exception of Away 3D, LLT MA did not outperform our approach and the differences in the F_1 values were significant (0.59 vs. 0.17 and 0.16 vs. 0.04, respectively). The difference on Away3D was small (F_1 values were 0.41 and 0.44, respectively). Similarly, LLT MA did not outperform LLT MA RE on the non-Java systems. The achieved results suggest that our approach and LLT MA RE are more independent from the kind of documents in the corpus. Since our approach was more accurate, we can then conclude that it is the best and can be applied without making any assumption on the mailing list and the programming language of the understudy system. The same did not hold for lightweight techniques based on regular expressions because they heavily rely on common conventions and intrinsic syntactical characteristics of the corpus [12].

C. Lesson Learned

The accuracy of our approach increased when e-mails contain a huge amount of text and the entity names are carefully chosen and naming conventions are used. Furthermore, when e-mails did not contain a huge amount of text, the application of BM25F on two or three fields did not produce noteworthy differences. Then, BM25F on header, body, and quote with the operator "AND" is the best alternative.

We experimentally observed that, in terms of accuracy, our approach outperformed on 5 out of 6 systems the lightweight technique that is more independent from the kind of e-mails in the corpus (i.e., LLT MA RE) [12]. To apply our approach, any assumption on the system understudy has to be made and any particular configuration setting is required. Therefore, our approach is easier to use than lightweight approaches and it is accurate enough to be worth the costs it may introduce in the corpus preprocessing and indexing phases. Furthermore, IR-based approaches, such as the one we introduce here, are more scalable. They are more efficient than lightweight techniques when the number of e-mails in the corpus increases. Finally, lightweight techniques return documents without any ranking: an e-mail either matches or not a regular expression. As a consequence, all the retrieved links have to be analyzed. In addition, incremental processes cannot be used to keep only relevant links (e.g., [29]).

1) Pieces of evidences: We distilled our findings and lesson learned into the following pieces of evidence (PoE):

PoE1. Accuracy increases when using class and package names as the queries;

PoE2. Applying our approach on three fields (i.e., header, body, and quote) improves the results when the corpus contains e-mails with a huge amount of text and the entity names are carefully chosen by developers;

PoE3. Using the "AND" operator leads to better results in terms of correctness, completeness, and accuracy;

PoE4. The corpus normalization by using stemming techniques reduces the accuracy of the recovered links;

PoE5. Our approach scales reasonable well also when the number of documents in the corpus increases;
PoE6. Our approach is more independent from the mailing list than lightweight approaches.

D. Threats to Validity

To comprehend the strengths and limitations of our study, we present here the threats that could affect the validity of the results and their generalization. Although our efforts in mitigating as many threats as possible, some threats are unavoidable. A possible threat is related to the used benchmark that is built on human judgement. The use of open source software represents another threat to validity. Although many large companies are using open source software in their own work or as a part of their marketed software, it will be worth replicating the study on real project. These replications will help us to confirm or contradict the achieved results. The instantiation of Query Formulation is another possible threat. We used class names or class and package names to compare our approach with those in [12].

V. Conclusion

We proposed, implemented [23], and evaluated an approach to recover links between e-mails and source code. The approach is based on text retrieval techniques combined with the BM25F probabilistic model. To assess the validity of our proposal, we conducted an empirical evaluation using a public benchmark [13]. Based on this benchmark, we performed a comparison between our approach and 8 baselines. The results indicated that our approach in many cases outperformed the IR-based baseline approaches and the lightweight techniques proposed in [12].

References

[1] E. B. Swanson, "The dimensions of maintenance," in *Proc. of International Conference on Software Engineering*. IEEE CS Press, 1976, pp. 492–497.

[2] A. V. Mayrhauser, "Program comprehension during software maintenance and evolution," *IEEE Computer*, vol. 28, pp. 44–55, 1995.

[3] S. Pfleeger and J. Atlee, *Software Engineering - Theory and Practice*. Pearson, 2006.

[4] A. De Lucia, F. Fasano, C. Grieco, and G. Tortora, "Recovering design rationale from email repositories," in *Proc. of the International Conference on Software Maintenance*. IEEE, 2009, pp. 543–546.

[5] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: The study of methods," *IEEE Trans. Software Eng.*, vol. 32, no. 1, pp. 4–19, 2006.

[6] A. Bacchelli, M. Lanza, and M. D'Ambros, "Miler: a toolset for exploring email data," in *Proc. of the International Conference on Software Engineering*. ACM, 2011, pp. 1025–1027.

[7] A. Bacchelli, T. Dal Sasso, M. D'Ambros, and M. Lanza, "Content classification of development emails," in *Proceedings of the 2012 International Conference on Software Engineering*. IEEE Press, 2012, pp. 375–385.

[8] N. Bettenburg, B. Adams, A. E. Hassan, and M. Smidt, "A lightweight approach to uncover technical artifacts in unstructured data," *Proc. of the International Conference on Program Comprehension*, vol. 0, pp. 185–188, 2011.

[9] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *ACM Trans. Softw. Eng. Methodol.*, vol. 16, no. 4, 2007.

[10] S. K. Sundaram, J. H. Hayes, A. Dekhtyar, and E. A. Holbrook, "Assessing traceability of software engineering artifacts," *Requir. Eng.*, vol. 15, no. 3, pp. 313–335, 2010.

[11] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering traceability links between code and documentation," *IEEE Trans. Software Eng.*, vol. 28, no. 10, pp. 970–983, 2002.

[12] A. Bacchelli, M. Lanza, and R. Robbes, "Linking e-mails and source code artifacts," in *Proc. of International Conference on Software Engineering*. ACM, May 2010, pp. 375–384.

[13] A. Bacchelli, M. D'Ambros, M. Lanza, and R. Robbes, "Benchmarking lightweight techniques to link e-mails and source code," in *Proc. of Working Conference on Reverse Engineering*. IEEE Computer Society, 2009, pp. 205–214.

[14] C. D. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press, 2008.

[15] S. Robertson, H. Zaragoza, and M. Taylor, "Simple bm25 extension to multiple weighted fields," in *Proc. of International Conference on Information and Knowledge Management*. ACM, 2004, pp. 42–49.

[16] S. Robertson and H. Zaragoza, "The probabilistic relevance framework: Bm25 and beyond," *Found. Trends Inf. Retr.*, vol. 3, pp. 333–389, April 2009.

[17] K. Y. Itakura and C. L. Clarke, "A framework for bm25f-based xml retrieval," in *Proc. of International Conference on Research and Development in Information Retrieval*. ACM, 2010, pp. 843–844.

[18] M. F. Porter, "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

[19] K. S. Jones, S. Walker, and S. E. Robertson, "A probabilistic model of information retrieval: development and comparative experiments," *Inf. Process. Manage.*, vol. 36, pp. 779–808, November 2000.

[20] J. R. Pérez-Agüera, J. Arroyo, J. Greenberg, J. P. Iglesias, and V. Fresno, "Using bm25f for semantic search," in *Proc. of the International Semantic Search Workshop*. ACM, 2010, pp. 2:1–2:8.

[21] J. Pérez-Iglesias, J. R. Pérez-Agüera, V. Fresno, and Y. Z. Feinstein, "Integrating the Probabilistic Models BM25/BM25F into Lucene," *CoRR*, vol. abs/0911.5046, 2009.

[22] L. Dolamic and J. Savoy, "When stopword lists make the difference," *J. Am. Soc. Inf. Sci. Technol.*, vol. 61, no. 1, pp. 200–203, Jan. 2010.

[23] L. Mazzeo, A. Tolve, R. Branda, and G. Scanniello, "Linking e-mails and source code with lasco," in *Proc. of the European Conference on Software Maintenance and Reengineering*. IEEE Computer Society, 2010.

[24] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering - An Introduction*. Kluwer, 2000.

[25] V. Basili, G. Caldiera, and D. H. Rombach, *The Goal Question Metric Paradigm, Encyclopedia of Software Engineering*. John Wiley and Sons, 1994.

[26] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society of Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[27] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations, Vol. 1*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2002.

[28] A. Marcus and J. I. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proc. of the International Conference on Software Engineering*. IEEE CS Press, 2003, pp. 125–137.

[29] A. De Lucia, R. Oliveto, and P. Sgueglia, "Incremental approach and user feedbacks: a silver bullet for traceability recovery," in *Proc. of the International Conference on Software Maintenance*. IEEE Computer Society, 2006, pp. 299–309.

IR based Traceability Link Recovery Method Mining

Takeyuki Ueda, Shinpei Ogata, Haruhiko Kaiya, Kenji Kaijiri
Shinshu University
Nagano, Japan
Email: ueda, ogata, kaiya, kaijiri@cs.shinshu-u.ac.jp

***Abstract*—Traceability link recovery is an important process in software development, and several researches are done, but the generality of adequate methods is not considered. The target of traceability link recovery includes several kinds of documents, so the adequacy of recovery methods depends on the characteristics of these documents, for example, an average similarity, a kind of document pair, document size, and so on. We propose the traceability link recovery method mining, which identifies a kind of adequate recovery method based on the characteristics of target documents by using knowledge base consisting of (a method, characteristics, and performance). This knowledge base shows which pair (a method, characteristics) is good at performance. Our target traceability link recovery method is IR based method, which is major method of automated traceability recovery. Some experiments based on the traceability reference data sets are done and the potential of our method is shown.**

Keywords-traceability; mining; information retrieval

I. Introduction

In experimental software engineering, especially in estimating software quality factors, several methods are proposed and the effectiveness of these methods is evaluated. However external validity of these evaluations is not validated, because there are a variety of target domains and these objective artifacts have a variety of characteristics. Therefore an experimental result for some artifacts is not applied to another artifacts, but validation of external validity is indispensable for application to real software artifacts. Wohlin [1] said that threats to external validity are conditions that limit our ability to generalize the results of our experiment to industrial practice.

Zhimin [2] proposed a method mining technique for error prone module prediction. In error prone module prediction, predictors are constructed based on training data by using some mining algorithm and software metrics, so these factors must be determined before applying prediction. There are many researches about this domain and several proposals for adequate algorithms, metrics, and training data have been done, but these results are not generally validated, that is, threats to validity about external validity is not solved. Zhimin [2] constructed a knowledge base about the adequate set for prediction, and by using this knowledge base, the adequate algorithm and the training data are estimated. Zhimin's main idea is to reuse the performance data based on the similarity of characteristics.

On the other hand, traceability link recovery is the important research topic in software maintenance. Traceability link is the relation between software artifacts, for example, requirement statements and design statements, design statements and source code, functional requirements and nonfunctional requirement. These links may be missed during development, so their reestablishment is needed. This reestablishment is the objective of traceability link recovery. Asuncion [3] categorizes traceability link recovery into two categories: retrospective traceability and prospective traceability. The former is automated approach and Information Retrieval(IR) based method is the representative and it recovers the traceability link based on the document automatically. The latter is semi automated or manual approach. The retrospective traceability is more available than the prospective traceability, but it is not so precise. So improvement and guarantee of preciseness are the main research topic in IR based traceability link recovery. There are many researches [4]–[12], and several methods are proposed, but the best method is not identified. The adequacy of method is considered to be dependent on the characteristics of target document pair.

In this paper, we propose the application of Zhimin's method to IR based traceability link recovery. In this case, the triple (a method instance, characteristics, performance) is training data, the tuple (a method instance, characteristics) is test data. A method instance is a certain combination of a variety of traceability link recovery techniques. Performance is the measure of how good this method instance is. Characteristics are factors which may affect the performance. We suppose that an average similarity, a kind of document pair, document size, and so on are candidates for the characteristics.

We also propose new cosine similarity, which reflects link semantics. Several experiments using reference data set provided in CoEST [13] are done and the effectiveness of new cosine similarity and our mining method is shown. Our main contributions are as follows:

- A traceability recovery method mining method is proposed.
- The accuracy of the selected traceability link recovery method is assured.
- Asymmetrical cosine similarity is proposed.

In Section 2, we describe the traceability link recovery problem and in Section 3, we discuss the application of the method mining model to traceability domain. In Section 4, we describe the experiment and the result, and show the effectiveness of this method. In Section 5, we consider threats to validity, and in Section 6, we compare with related researches. We conclude our paper and consider future work in Section 7.

II. IR BASED TRACEABILITY LINK RECOVERY

Traceability link means the relation between some software components within several software documents. A software project has several kinds of documents and these documents consist of several components; for example, a software project has requirement document, design document, source code and test document, and each of these documents have their own components (the requirement document consists of many requirement statements and the source code consists of many class files).

There are several approaches about traceability link recovery [4] and the most possible method is IR based method. In IR based method, each component is modeled as a term vector, and the similarity between components is measured by using the cosine similarity of these term vectors. Traceability link will be identified by using these similarity values. These term vectors are aggregated into a term document matrix. There are several variations of construction methods of a term document matrix:

1) Term extraction and preprocessing: Stemming, stop word, Camel case
2) Kind of value for term vector: True/False, frequency, term frequency-inverse document frequency(TF-IDF)
3) Link candidate judgment method by using similarity value: threshold value (top n%) or rank (top n pairs)
4) Modification of term document matrix: Latent Semantic Indexing(LSI)

These are traditional variations for IR based method. We consider a further variation, which is specific for software link recovery, asymmetric cosine similarity. The cosine similarity treats each component symmetrically, but several kinds of relations are proposed in software traceability [14], [15], for example, Ramesh [14] proposed the following four kinds of link:

- Satisfaction link
- Evolution link
- Rationale link
- Dependency link

These relations are not necessarily symmetric, so we define asymmetrical cosine similarity as follows:

$$X \times Y/(|X| * |Y|) \qquad (1)$$

if $X_i == 0$ then the corresponding Y_i is not considered.
where X and Y are term vectors and X=(X_1,,,,X_n) , Y=(Y_1,,,,Y_n)

Asymmetrical similarity considers only how much X is covered by Y, for example, X=(0,0,1,1,0) and Y=(1,0,1,1,0), then symmetrical similarity(X,Y) = 2/(sqrt(2)*sqrt(3))=0.816, and asymmetrical similarity(X,Y)=2/(sqrt(2)*sqrt(2))=1.0.

We apply this variation as the 5th variation.

There are several other variations, for example, the granularity of components, ontology, etc. The treatment of these alternatives is the future research theme.

Each traceability recovery method selects one alternative from each variation. The following is an example:

- stemming is used
- stop word is eliminated
- camel case word is decoupled
- value of term document matrix is TF-IDF
- link candidate is judged with threshold value (0.3)
- LSI is applied
- symmetrical cosine similarity is applied

We call these alternatives as method instances. Selection of adequate method instances for each data is the main target of our research.

We afford the following research questions:

- RQ1: Is it possible to identify the adequate method instance for each project data?
- RQ2: Is it possible to assure the accuracy of the selected method instance?
- RQ3: Is the asymmetric similarity is effective for traceability link recovery?

III. TRACEABILITY LINK RECOVERY METHOD MINING

We show the traceability link recovery method mining in Figure 1. In order to do traceability link recovery, an adequate method instance has to be identified, and we supposed that the adequacy is dependent on the characteristics of target documents, so it must be possible to identify the adequate method instance candidate by using these characteristics. We use data mining approach proposed by Zhimin [2] for this identification. For this purpose we need to select the adequate characteristics. In this paper, we use CoEST [13] data set as a reference data set. Each document consists of two component sets and link between these component. We select the following characteristics which can be extracted from these documents:

- Average similarity
- Number of components
- Total term count
- Used language
- Type of document relation

These characteristics may be insufficient and the adequacy needs to be further considered.

As shown in Figure 1, the following training data and test data are needed:

- Training data: (a method instance, characteristics, performance)
- Test data : (a method instance, characteristics)

Performance is transformed into true/false value based on the traceability link criterion which is defined by using precision and recall. There are a few reports about the traceability link criterion. Hayes [16] described that adequate recall value is from 60 to 69% and adequate precision value is from 20 to

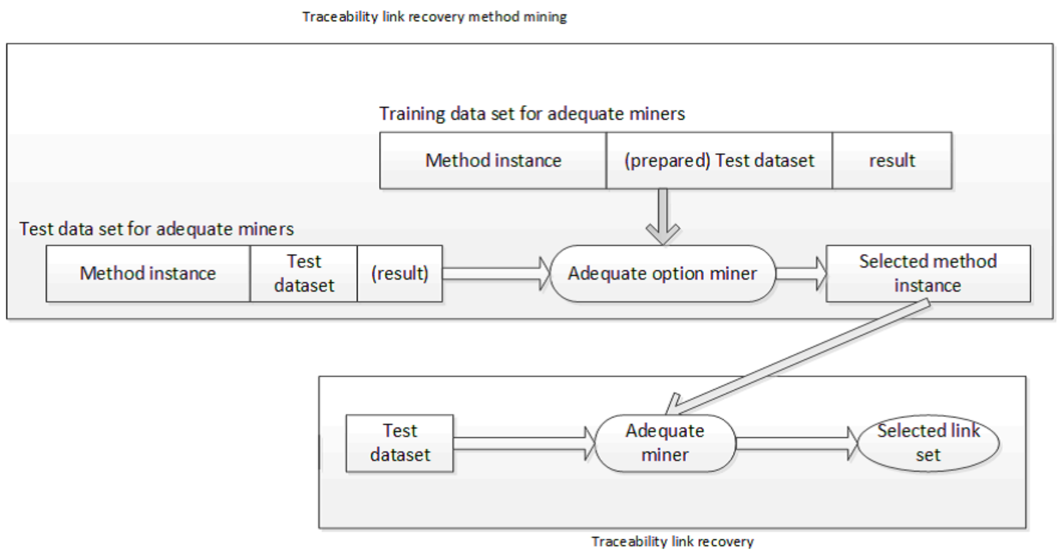


Fig. 1. Outline of traceability link miner selection

29% based on enterprise experiences. We use these values as objective values, and adjust these values based on conditions.

We show the procedure for traceability link recovery method mining.

1) For each method instance and each data in reference data set
   a) Do traceability link recovery
   b) Based on the traceability link criterion, judge the adequacy of recovery result
2) Training data is generated
3) Do method mining by using training data
4) If the identification performance is below the identification criterion, lower the traceability link criterion and repeat from step 2
5) By using this data, we construct a method miner
6) Construct test data
7) Do method mining by using (method minor, test data) pair
8) If no method is mined, lower the traceability link criterion and repeat from step 2
9) Do traceability link recovery by using the selected method instance

Step 3 and 4 are for the examination of adequacy of the selected training data. If the traceability link criterion becomes too small in step 8, it means that reasonable traceability link recovery is impossible for this test data. There are the following reasons:

- Training data is inadequate for test data.
- A variety of method instances are insufficient.
- Method mining method is insufficient.
- The quality of the document is too low.

We define two criteria:

- Traceability link criterion
  There must be adequate accuracy value in traceability link recovery, and it is the traceability link criterion. We use Hayes proposed value for this criterion, that is, $precision > 0.3$ and $recall > 0.7$, but there are several cases for this adequacy, so we may adjust this criterion. We further call the pair (precision, recall), which is the result of traceability link recovery, as traceability link recovery performance. We call a method instance, which satisfies this criterion, as a candidate method instance.
- Identification criterion
  The selected method instance needs to assure the satisfaction of the given traceability link criterion. We use the following precision for this purpose

$$\frac{(\mid CMIS \mid) \cap (\mid SMIS \mid)}{\mid SMIS \mid} \tag{2}$$

where CMIS is a set of candidate method instances, and SMIS is a set of selected method instances. This criterion is computed for each project, that is, a pair of documents. For example, if the identification criterion is 0.8, then 80% of the selected method instances are supposed to satisfy the traceability link criterion. We call the result of traceability link recovery method mining as identification performance. It only shows possibility, that is, the satisfaction of the traceability link criterion is only exemplified using training data, so if the test data has similarity with the training data, this possibility is high, but if the test data has no similarity, then this possibility is low.

IV. Experiments

We used the subset of the reference data set provided by CoEST [13]. The details of the used data sets are shown in Table I. The third column presents the numbers of each component (source component and destination component), the fourth column contains the average number of correct

TABLE I
REFERENCE DATA SET

project	document pair	# of components	average # of links	# of candidate method instances
eAnci	UC_CC	140,55	3.10	0
Gantt	high_low	17,69	4.00	0
SMOS	UC_CC	67, 100	15.58	0
WV_CCHIT	Requirements-Regulatory code	116, 1064	5.06	0
EasyClinic	CC_TC	47,63	4.34	0
EasyClinic	ID_CC	20, 47	3.45	89
EasyClinic	ID_TC	20, 63	4.15	37
EasyClinic	ID_UC	20, 30	1.30	13
EasyClinic	TC_CC	63, 47	3.24	0
EasyClinic	UC_CC	30, 47	3.10	47
EasyClinic	UC_ID	30, 20	0.87	15
EasyClinic	UC_TC	30, 63	2.10	16
Waterloo grp01	high,low	58, 26	0.52	0
Waterloo grp02	high,low	42, 13	1.24	0
Waterloo grp03	high,low	70, 28	1.34	0
Waterloo grp05	high,low	54, 30	0.87	2
Waterloo grp06	high,low	39, 21	1.41	0
Waterloo grp08	high,low	85, 22	1.08	0
Waterloo grp09	high,low	30, 19	1.77	0
Waterloo grp10	high,low	76, 8	0.91	0
Waterloo grp11	high,low	79, 9	0.89	0
Waterloo grp13	high,low	43, 8	0.72	0
Waterloo grp14	high,low	46, 5	0.72	24
Waterloo grp15	high,low	69, 27	1.35	0
Waterloo grp17	high,low	57, 7	0.89	0
Waterloo grp18	high,low	53, 8	0.66	78
Waterloo grp19	high,low	61, 15	2.03	0
Waterloo grp20	high,low	93, 14	1.49	0
Waterloo grp21	high,low	36, 26	1.14	25
Waterloo grp23	high,low	32, 20	1.06	12
Waterloo grp24	high,low	51, 29	1.10	0
Waterloo grp30	high,low	48, 20	0.73	0
Waterloo grp32	high,low	86, 21	1.57	0
Waterloo grp33	high,low	65, 11	0.94	0
Waterloo grp34	high,low	28, 16	0.64	0

links, and the fifth column contains the number of method instances, which identify link candidates with $precision > 0.3$ and $recall > 0.7$.

The detailed experimental results are too large, so we store the results in http://cwww.cs.shinshu-u.ac.jp/ICSEA/ and show only the summarized results.

We used seven threshold values (10%, 20%, 30%, 40%, 50%, 60%, 70%) and four rank values (5, 10, 15, 20) with the five kinds of variation described in Section II, so the total number of method instances is 1056.

We did three experiments by using the data mining tool Weka [17] in order to evaluate the effectiveness of our method.

A. Experiment 1: Traceability link recovery evaluation for the 35 data

We did 1056 runs (method instances) for each data: total 36960 (1056×35) runs. Each run calculates candidate link set and the accuracy is evaluated by using the given answer link set.

TABLE II
THE RESULT OF TRACEABILITY LINK RECOVERY

	precision	recall	f-measure
average	0.196	0.520	0.171
standard deviation	0.228	0.376	0.128

We evaluated the effectiveness of each method instance for each data. We show the number of candidate method instances, whose traceability link performance satisfies the traceability link criterion, in Table I and the statistic values in Table II. In this experiment, the traceability link criterion is $precision > 0.3$ and $recall > 0.7$. The deviation of the number of candidate method instances are large, that is, in 24 out of 35 data, the number of candidate method instances is zero, but EasyClinic ID_CC has 89 candidate method instances and Waterloo grp18 has 78 candidate method instances. The standard deviations of performance values (precision, recall, f-measure) are also large, so there must be adequacy of method instances for each data set. We show the scatter plot diagram in Figure 2. The horizontal axis is the index of each method instance and the vertical axis is the number of occurrences in the top 3 method instances with f-measure.

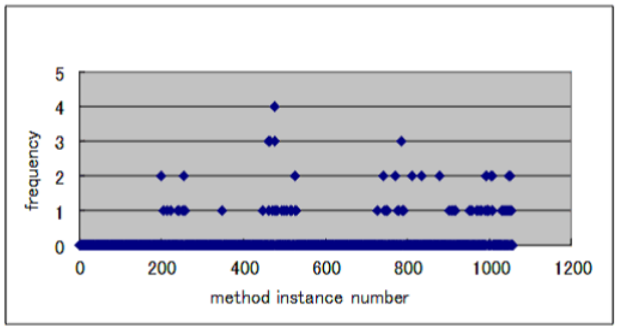


Fig. 2. Scatter plot diagram top 3 method instances

This figure also shows that there is no unique method instance, which has the best performance, that is, plots are dispersed. The O and P column of exp1.xlsx in http://cwww.cs.shinshu-u.ac.jp/ICSEA/ show the traceability link recovery result with f-measure. F-measure for asymmetrical similarity is better than symmetrical similarity for 14 data, but worse for only one data, so in some cases, asymmetrical similarity is better method.

From experiment 1, the necessity of the adequate method instance selection and the effectiveness of asymmetrical similarity become clear.

B. Experiment 2: Cross Validation

Experiment 2 is the traceability link recovery method mining experiment. First we try cross validation in order to evaluate the possibility of our proposed method. We integrated the results of the experiment 1 into one data (weka format file) and did 10 fold cross validation by using several algorithms.

TABLE III
THE IDENTIFICATION PERFORMANCE (CROSS VALIDATION)

	algorithm	p=0.5 r=0.7	p=0.3 r=0.7	p=0.2 r=0.6	p=0.1 r=0.5
precision	J48	0.92	0.87	0.89	0.96
precision	Naive Bayes	0.04	0.08	0.15	0.37
precision	Logistic	0.62	0.58	0.55	0.64
precision	Random Forest	0.72	0.75	0.80	0.95
recall	J48	0.54	0.68	0.82	0.99
recall	Naive Bayes	0.69	0.78	0.75	0.96
recall	Logistic	0.09	0.06	0.08	0.46
recall	Random Forest	0.54	0.62	0.74	0.95

We show the identification performance for several traceability link criteria in Table III. Precision and recall in the first column mean the precision and recall of identification performance, and p and r in the first row mean the precision and recall of traceability link recovery performance.

We got acceptable precision values for each traceability link criterion, so the potential of our method becomes clear, but the dependency on algorithms is very high. Even for the objective traceability link criterion ($precision > 0.3$ and $recall > 0.7$), the precision is 0.87 in J48 and 0.75 in Random Forest, but 0.08 in Naive Bayes. These experiments only show the average performance, and in order to show the possibility for each data, the next experiment is needed.

C. Experiment 3: Identification performance check for each data

Our training data contains 1056 data for each project, that is, 36960 data. In cross validation these 36960 data is divided into ten subsets, and the combination of nine subsets is used as training data and the remaining one subset is used as test data, so the training data includes many data whose project is the same as the data in the test data. In the objective traceability link criterion, the number of candidate method instances is 358, that is only 1% (358/36960), and the number of projects which have candidate method instance is 10 (total is 35), so the bias between training data and test data may exist. As the result of this condition, identification performance may be overestimated, so we evaluate the performance for each project.

We constructed training data from N-1 data, and test data from the remaining data. In our experiment, N=35, so we constructed 35 pairs. We show the summarized result in Table IV and the detailed result in http://cwww.cs.shinshu-u.ac.jp/ICSEA/exp3.xlsx. 11 kinds of traceability link criterion are tested and the results are shown in Table IV.

For the objective traceability link criteria ($precision > 0.3$ and $recall > 0.7$), identification performances are low, that is, only one data satisfies the identification criterion ($precision > 0.7$). In almost all data, precision is zero even for very low traceability link criteria. We can get only the reasonable value, that is, identification performance ($precision > 0.7$)

TABLE IV
THE IDENTIFICATION PERFORMANCE (BY PROJECT VALIDATION)

precision	recall	# of data in which the traceability link recovery performance satisfies the criterion	# of projects whose identification precision is greater than 0.7	# of projects whose identification precision is greater than 0.5
0.7	0.7	4	0	0
0.6	0.7	20	0	0
0.5	0.7	87	1	1
0.4	0.7	164	0	1
0.3	0.7	358	1	1
0.3	0.6	655	2	4
0.2	0.7	1195	2	5
0.2	0.6	1873	3	4
0.1	0.5	9807	21	26
0.1	0.4	11238	23	27
0.05	0.5	16985	26	31

and traceability link recovery performance ($precision > 0.3$ and $recall > 0.7$) in the case of EasyClinic ID_CC, and traceability link recovery performance ($precision > 0.3$ and $recall > 0.6$) in the case of EasyClinic UC_ID.

This result can not show the potential of our method, so we did further experiments in order to consider the reasons and the possibility to improve identification performance.

We considered the following reasons:

- The number of candidate method instances is too small. As shown in Table I, the number of candidate method instances is too small compared with the number of tested method instances (1056) and deviation is large. In the case of $precision > 0.3$ and $recall > 0.7$, 11 out of 35 projects have the value zero, and further the percentage of the candidate method instances are low, that is, the most high case is for EasyClinic ID_CC and the value is 8% (89/1056). Training of succeeful pair based on a few succeeded data is very difficult, so better traceability link recovery methods or customization are needed in order to augument the number of candidate method instances.
- Each data has special link characteristics. For example, in SMOS the average number of link is especially larger than others, and in WV_CCHIT the number of destination components is larger than others, so the adequacy of method instance is a little different from each other and as the result of this difference the training data generated from such inadequate data becomes inadequate.
  We did method mining experiments for SMOS and WC_CCHIT by using selected training data in order to evaluate the matching of a training data and a test data. The detailed results are shown in http://cwww.cs.shinshu-u.ac.jp/ICSEA/ and summarized result is in Table V, which shows only the top two results. “all” in training data means the original experiment 3. # of T means the number of candidate method instances. The first 4 rows

show that the identification performance is 0.145 for the case of using all data, but is 0.687 (0.466) for the case of using EAnci_UC_CC (Waterloo grp09). This result shows that the adequate training data improves the identification performance, that is, the identification performance by using adequate training data is larger than the case of the integrated training data. This adequacy may depend on the data characteristics, but the identification of these characteristics is now an open problem.

V. Threats to Validity

Regarding the internal validity, the variation of alternatives and the characteristics of documents are not sufficient. This research is still ongoing, and the main objective of this paper is to show the potential of the proposed method, so the result is not sufficient. The following method options and characteristics are to be considered:

- Further method options
  - Latent Dirichlet Allocation(LDA) and/or ontology application [18]
  - Granularity factor (How to divide a document into components)
  - What kind of term is to be used
- Document (pair) characteristics
  - Variance of similarity
  - Refined classification of document pair (in Easy-Clinic data set, there are four kinds of documents)
  - Language information ([7] shows that the English version and the Italian version have different result)
  - Development member variation
  - Estimated value of the number of traceability links

Regarding the external validity, the used data sets are not sufficient. We did not test all of the CoEST data set. Also real software documents have many variation (plain text, office document, CAD based document, etc) and have language problems. Granularity of components is the important factor, but CoEST data set is already separated as link unit, so other granularity is not tested. Our result only shows the availability of proposed method ming, so in order to apply to some real softwares, the corresponding knowledge base needs to be constructed.

VI. Related Research

There are many researches about traceability recovery and there are several methods categories: rule base, IR base, and format base. IR based method has wide availability because it entails no constraint to developers, but as the result of this weak constraint, accuracy is not so good. For IR based method, in order to improve accuracy, several methods are proposed and evaluated [4]–[12], [19]. Lucia [8] evaluated the effect of term identification methods. Wiese [10] considered the stemming effect, and Mahmoud [11] considered how to construct a term-document matrix. Capobianco [6], [7] and Lormans [20] compared several IR based traceability recovery methods: Jenson-Shannon Method(JS), Vector Space Model(VSM), LSI, LDA. Lormans said that the adequacy of these methods are dependent on the kind of document.

Several criteria are used for traceability link candidate judgment. Lormans [20] compares the following five methods and concludes that the adequacy is dependent on the kind of document:

- Cut point: we select top k links with similarity value
- Cut percentage: we select k percentage of the ranked list
- Constant threshold: we select those links that have a similarity measure greater than k
- Variable threshold: we select those links that have a similarity measure greater than k, where k is calculated according to a percentage of the total similarity measures
- Scale threshold: we select links according to k = c * MaxSimilarity where $0 \leq c \leq 1$

There are many researches which evaluate several methods, but external validity is not considered sufficiently, so engineers cannot select/use the adequate method for their projects.

VII. Conclusion and Future Work

We proposed traceability link recovery method mining in order to select adequate method instances and to assure the traceability link criterion. Our experimental result shows the potential of our proposed method, but the accomplished identification performance is not sufficient. Regarding the research questions (RQ1 and RQ3), the answer is yes for some projects, but no for the other projects. It shows the heavy project dependency. In order to resolve this dependency, we need to improve both traceability link recovery performance and identification performance. For the former improvement, the following alternatives are planned:

1) LDA and statistic model
2) Candidate link judgment. Lormans [20] defined five judgment methods. We only used two, so the remaining three methods are to be evaluated.
3) There may be several categories about the document link properties, so the similarity functions which are adequate for these links are needed.

For the latter improvement, the followings are planned;

1) There exist several reference data, which are not evaluated, so we do further experiments using those data and consider the matching of training data and test data.
2) The used characteristics are not sufficient, so we consider the characteristics which are more related with document link properties.

Regarding the research question (RQ2), the answer is almost yes, but the relation between the effectiveness and the document characteristics is not clear. Further consideration about this relation is needed.

References

[1] C. Wohlin, P. Runeson, M. Hoest, M. C. Chisson, B. Reqnell, and A. Wessln, Experimentation in Software Engineering. Springer, 2012.
[2] Z. He, F. Shu, Y. Yang, M. Li, and Q. Wang, “An investigation on the feasibility of cross-project defect prediction,” Automated Software Engineering, vol. 19, no. 2, 2012, pp. 167–199.

TABLE V
THE IDENTIFICATION PERFORMANCE FOR SMOS AND WV_CCHIT

test data	precision	recall	# of T	training data	precision	recall	# of T	identification performance (precision)
SMOS	0.05	0.3	137	all	-	-	-	0.145
SMOS	0.05	0.3	137	EAnci_UC_CC	0.05	0.4	187	0.687
SMOS	0.05	0.3	137	Waterloo grp09	0.1	0.5	164	0.466
SMOS	0.05	0.2	279	all	-	-	-	0.506
SMOS	0.05	0.2	279	EAnci_UC_CC	0.05	0.3	296	0.781
SMOS	0.05	0.2	279	EasyClinic CC_TC	0.1	0.5	262	0.665
WV_CCHIT	0.05	0.3	136	all	-	-	-	0.111
WV_CCHIT	0.05	0.3	136	EasyClinic ID_CC	0.2	0.7	164	0.402
WV_CCHIT	0.05	0.3	136	waterloo grp23	0.2	0.6	110	0.359
WV_CCHIT	0.05	0.2	258	all	-	-	-	0.376
WV_CCHIT	0.05	0.2	258	EasyClinic CC_TC	0.1	0.5	262	0.607
WV_CCHIT	0.05	0.2	258	EasyClinic ID_CC	0.2	0.6	224	0.550

[3] H. U. Asuncion, A. U. Asuncion, and R. N. Taylor, “Software traceability with topic modeling,” in ICSE, ACM, 2010, pp. 95–104.

[4] A. Abadi, M. Nisenson, and Y. Simionovici, “A traceability technique for specifications,” in ICPC, 2008, pp. 103–112.

[5] M. Borg, K. Wnuk, and D. Pfahl, “Industrial comparability of student artifacts in traceability recovery research,” in CSMR, 2012, pp. 181–190.

[6] G. Capobianco, A. D. Lucia, R. Oliveto, A. Panichella, and S. Panichella, “On the role of the nouns in ir-based traceability recovery,” in ICPC, 2009, pp. 148–157.

[7] G. Capobianco, A. D. Lucia, R. Oliveto, A. Panichella, and S. Panichella, “Traceability recovery using numerical analysis,” in WCRE, 2009, pp. 195–204.

[8] A. D. Lucia, M. D. Penta, and R. Oliveto, “Improving source code lexicon,” IEEE Tr. on S.E., vol. 37, no. 2, 2011, pp. 205–227.

[9] A. D. Lucia, F. Fasano, R. Oliveto, and G. Tortora, “Recovering traceability links in software artifact management systems using information retrieval methods,” TOSEM, vol. 16, no. 4, 2007, pp. 1–50.

[10] A. Wiese, V. Ho, and E. Hill, “A comparison of stemmers on source code identifiers for software search,” in ICSM, 2011, pp. 496–499.

[11] A. Mahmoud and N. Niu, “Source code indexing for automated tracing,” in TEFSE, 2011, pp. 3–9.

[12] R. Oliveto, M. Gethersy, D. Poshyvanyky, and A. D. Lucia, “On the equivalence of information retrieval methods for automated traceability link recovery,” in ICPC, 2011, pp. 68–71.

[13] “Center of excellence for software traceability.” http://www.coest.org/. 08.01.2013.

[14] B. Ramesh and M. Jarke, “Toward reference models for requirements traceability,” IEEE Tr. on S.E, vol. 27, no. 1, 2001, pp. 58–93.

[15] W. Jirapanthong and A. Zisman, “Supporting product line development through traceability,” in APSEC, 2005, pp. 1–9.

[16] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, “Advancing candidate link generation for requirements tracing: The study of methods,” IEEE Tr. on S.E., vol. 32, no. 1, 2006, pp. 4–19.

[17] I. H. Witten and E. Frank, Data Mining - Practical Machine Learning Tools and Techniques. Morgan Kaufman, 2005.

[18] Y. Zhang, R. Witte, J. Rilling, and V. Haarslev, “An ontological approach for the semantic recovery of traceability links between software artifacts,” Software, IET, vol. 2, no. 3, 2008, pp. 185–203.

[19] A. Marcus, “Recovery of traceability links between software documentation and source code,” International Journal of Software Engineering and Knowledge Engineering, vol. 15, no. 5, 2005, pp. 811–836.

[20] M. Lormans and A. van Deursen, “Can lsi help reconstructing requirements traceability in design and test?,” in CSMR, IEEE, 2006, pp. 1–10.

Towards Identifying the Factors for Project Management Success in Global Software Development: Initial Results

Mahmood Niazi [a, b], Sajjad Mahmood [a], Mohammad Alshayeb [a], Abdul Majid Qureshi [a], Kanaan Faisal [a]
[a] Department of Information and Computer Science, King Fahd University of Petroleum and Minerals, Saudi Arabia
[b] Faculty of Computing, Riphah International University Islamabad, Pakistan
{mkniazi, smahmood, alshayeb, g201105290, kanaan}@kfupm.edu.sa

Narciso Cerpa
Faculty of Engineering, Universidad de Talca, Chile
ncerpa@utalca.cl

Abstract* – Global Software Development (GSD) is a collaborative development where one company (client) contracts out all or part of its software development activities to another company (vendor), which provides services in return for remuneration. In today's world of high cost commitments and limited budgets, GSD provides a viable option for developing lower cost product with a relatively better quality. However, this comes at the cost of overcoming various challenges of managing a project, which is geographically distributed. The objective of this paper is to identify a set of factors that contributes to the success of project management in GSD. We have performed a Systematic Literature Review (SLR) by applying customized search strings derived from our research question. We have identified success factors, such as organizational structure, project managers' skills, communication, requirement specification, cultural awareness, and trust building. Our ultimate aim is to develop a model in order to measure organizations' project management readiness for GSD activities.

***Keywords-** Global software development; Software Project Management; Systematic Literature Review; Empirical Study.*

I. INTRODUCTION

Low cost software development has always been the preamble of many organizations. If this low cost development comes with the added advantage of the high quality product then it adds to increase long term benefits for the organizations [1]. The search for the high quality and low cost development has led many organizations to use Global Software Development (GSD) model [2]. GSD is the process where a company either has its software developed by geographically distributed teams or contracts all or part of its software development activities in return for remuneration [3]. Majority of companies have adopted GSD to gain several perceived benefits, such as reduced software development time, access to skilled human resources at relatively low cost and increase in product quality [2,4]. GSD is significantly changing the economic drivers of software industry due to round the clock availability of skilled personals at lower cost.

Despite GSD benefits, the cultural differences associated with geographically distributed teams and different time-zones have caused problems for GSD-based projects [5,6]. The key GSD challenges are: lack of client involvement, hidden costs, lack of trust among the outsourcing companies, lack of coordination mechanisms and communication issues [5,6,7]. One of the major challenges is that many organizations endorse global contracts prior to testing their project management readiness for the global development activity. Despite the importance of this issue, little research has been carried out to improve organizations project management readiness for GSD. We believe that a better understanding of the factors associated with successful GSD project management can assist in improving organizations' project management readiness for GSD projects.

The advances in GSD have not been matched by equal advances in the development of new research and practices in academia and software industry, which has resulted in a gap between the software industry and academia. The up-to-date research in this area can help to fill this gap.

In this paper, we aim to identify success factors via SLR that impact project management in GSD projects. Identifying these factors will assist GSD organizations in better preparing for challenges associated with project management. Our long term research goal is to develop a global project management readiness framework to assist software development organizations in measuring and improving their project management readiness prior to starting global activities. To achieve this, we intend to address the following research question in this paper:

RQ: What factors are essential for the success of project management in GSD?

The organization of this paper is as follows: Section 2 provides the background. The research methodology is explained in Section 3. In Section 4, we present and discuss the initial results. Finally, we present the conclusion in Section 5.

II. BACKGROUND

Client organizations benefit from offshore outsourcing because vendors in developing countries (offshore vendors) typically cost one-third less than onshore vendors and even less when compared with in-house operations [4]. Among many other reasons for outsourcing, generally, client organizations outsource their software development work to offshore locations to: gain cost and quality advantages, improve their skills' access leading-edge technologies, and focus on their core competencies [8]. It is professed that offshore outsourcing vendors can add significant value to their clients' supply chains [9]. Conversely, quite apart from the outsourcing benefits, there are many risks in the outsourcing process, such as temporal incompatibility, cultural differences and hidden costs [5,6]. IT Week magazine reported that eight out of every ten firms that outsourced their software development project to an offshore vendor faced major problems due to insufficient preparation and poor management by both client and vendor organizations [10].

There are many reasons for these problems [10,11]. One of the major issues is that many clients endorse global contracts with their vendors prior to testing their project management readiness for the global activity [1]. For example, a recent SLR concluded that the Global Software Engineering field is still nascent and comparatively very few empirical studies have been conducted, which can help to resolve the problems in this domain [12]. Understanding issues related to organization's global project management readiness will help to ensure the successful outcome of projects and to maintain long lasting relationships between clients and vendors in different geographical locations [3]. Hence, in this paper we conduct a SLR to identify project management challenges in GSD projects. The collected data focuses on factors that are essential for the success of project management in GSD.

III. RESEARCH METHODOLOGY

A SLR process [13] was used as the approach for data collection. SLR is a defined and methodical process of identifying, assessing, and analyzing published primary studies in order to investigate a specific research question. Systematic reviews differ from ordinary literature surveys in being formally planned and methodically executed. In finding, evaluating and summarizing all available evidence on a specific research question, a systematic review may provide a greater level of validity in its findings than might be possible in any one of the studies surveyed in the systematic review. A systematic review protocol was written to describe the plan for the review. The major steps in our methodology are:

- Constructing search strategy and then perform the search for relevant studies.
- Perform the study selection process.
- Apply study quality assessment.
- Extract data and analyze the extracted data.

This paper focuses on identifying the factors for successful project management in GSD, and therefore, we intended to address the following research question:

RQ: What factors are essential for the success of project management in GSD?

A. *Search Strategy*

The search strategy has been based on following steps:

- Derive the major terms from Population, Intervention, and outcome.
- Find synonyms and similar spellings of the derived terms obtained above.
- Verify these terms in various academic databases
- Use Boolean operators (AND operator is used to connect major terms. OR operator is used to connect synonyms and similar spellings).

Based on the above search strategy, we have constructed the following search terms:

- *POPULATION:* Global Software Development (GSD) organizations.
- *INTERVENTION:* Project management success factors.
- *OUTCOME OF RELEVANCE:* Factors for successful project management of GSD.
- *EXPERIMENTAL DESIGN:* SLRs, empirical studies, theoretical studies and expert opinions.

We test our terms in various academic databases and the following terms show potential relevance to the topic:

- *GLOBAL SOFTWARE DEVELOPMENT:* Global Software Development OR GSD OR distributed software development OR multisite software development OR multi-site software development OR global software teams.
- *PROJECT MANAGEMENT:* Software Project Management OR Software Development Management OR Software Process Management.
- *FACTORS:* Factors OR causes OR agents OR elements OR aspects OR determinants OR constituents OR ingredients.
- *CONTRIBUTE:* Contribute OR furnish OR provide OR supply.
- *SUCCESS:* Success OR advance OR progress OR favorable OR effective.
- *IMPLEMENT:* implement OR apply OR utilize OR device OR mechanize.
- *PRACTICE:* procedure OR form OR method OR perform OR exercise.

The final search string is a combination as follows:

{Global Software Development OR GSD OR distributed software development OR multisite software development OR global software teams} AND {Factors OR causes OR agents OR elements OR aspects OR determinants OR constituents OR ingredients} AND {Contribute OR furnish OR provide OR supply} AND {Success OR advance OR progress OR favorable OR effective}

B. *Digital Libraries used*

Based on the available access, the following digital libraries were used:

- ACM Digital Library.
- IEEE Explore.
- Science Direct.
- Google Scholar
- ISI Web of Science.
- Springer Link.

C. *Inclusion and Exclusion Criteria*

Since these libraries differ in their search mechanism and capability, we tailored our search strings accordingly.

The following inclusion criteria were used:

- Conference Proceedings, Magazines, and Journals published after 1980.
- Papers published in any of the primary or secondary resources mentioned previously.
- Studies focus on answering our research question.

The following exclusion criteria were used:

- Papers published before 1980 are excluded since Internet starts after that date.
- Manuscripts written in non-English language are excluded.
- Technical reports and white papers are excluded.
- Graduation projects, master theses and PhD dissertations are excluded
- Textbooks whether in print or electronic are excluded from this systematic review.

D. *Selection Process*

The planned selection process had two parts: an initial selection from the search results of papers that could plausibly satisfy the selection criteria, based on a reading of the title and abstract of the papers, followed by a final selection from the initially selected list of papers that satisfy the selection criteria, based on a reading of the entire papers. In order to reduce the researcher's bias, we have performed the inter-rater reliability test.

TABLE I. QUALITY ASSESSMENT

Criteria	Notes

Are the findings and results clearly stated in the paper?	Yes =1
	No =0

Is there any empirical evidence on the findings?	Yes =1
	No =0

Are the arguments well- presented and justified?	Yes =1
	No =0

Is the paper well referenced?	Yes =1
	No =0

For any paper to pass the selection process, a quality assessment was done. Four quality criteria were prepared as shown in Table I. We have finally selected 118 articles, which meet our inclusion and quality criteria.

E. *Data extraction*

From the finally selected papers, we have extracted data in order to address our research question. The following data was extracted from each paper: publication type, authors, publisher, publication name, publication date, organization size, project size, success factors and best practices.

IV. INITIAL RESULTS AND DISCUSSION

The total number of articles retrieved after using the search terms in the five electronic databases are shown in Table II. After the initial round of screening by reading the title and abstract, 292 studies relating to five different electronic databases were selected. After full text readings in the second screening, 118 primary studies were finally selected.

We have grouped the papers found through SLR into three broad study strategies, which are commonly used in the empirical software engineering, as shown in Table III. Most of the articles have used survey research method. These study strategies were initially identified by one researcher during the data extraction process. However, second researcher has validated these study strategies.

TABLE II. SEARCH EXECUTION

Resource	Total Results	Initial Selection	Final Selection
IEEE Xplore	639	238	92
ACM	29	14	7
Science Direct	27	10	4
Springer Link	28	13	7
John Wiley	31	17	8
Total	754	292	**118**

TABLE III. STUDY STRATEGIES USED

Study Type	Count
Case Studies	43
Systematic Literature Reviews (includes literature reviews)	23
Survey (includes interviews, experience reports, Delphi studies)	52
Total	118

In Table IV, we show the countries where research was conducted for the papers included in our SLR study. Not surprisingly, the maximum number of studies (a total of 43) was carried out in the United States. This might be due to the fact that most of the multinational giants in the United States prefer GSD mode of development in collaboration with third world countries like India and China.

On the other hand, many studies have also been carried out in eastern countries like India, China, and Pakistan as these countries are providing vendor services in GSD projects. Other geographic locations include Netherlands, Ireland, and United Kingdom, where the communication is carried out in English language and culturally these countries are more or less similar.

TABLE IV. STUDY COUNTRIES

Country	Count	Country	Count
Australia	4	New Zealand	1
Brazil	5	Latvia	3
Canada	2	Malaysia	2
China	5	Netherlands	5
Singapore	1	Croatia	1
Finland	5	Pakistan	1
Germany	3	South Africa	1
Hawaii	4	Spain	2
India	10	Sweden	1
Iran	1	Saskatoon	1
Ireland	9	United Kingdom	5
Berlin	3	USA	43

In total, 18 factors that lead to the success of project management in GSD projects have been identified as shown in the table V. Initially, we have identified 29 success factors. After a few iterations, these factors were reduced to 18 as many factors had similar meaning. These factors have been arranged in decreasing order of their frequencies (frequency here is a measure of the number of times each factor has been suggested/mentioned in the selected study).

In our study, the most common project management success factor in GSD is the 'organizational structure' (62%). The organizational structure includes the entire dynamics of the GSD is risk-prone and hence require special as processes and people. GSD organizations follow different structures in order to successfully manage global projects as shown in Figure 1 [14]. Figure 1 contains an organizational structure to develop and implement a new software tool in five countries with all program managers located in the UK. The software is developed in three countries (project manager is located in the UK and the project team members are located in the UK, Singapore and Mexico). The pilot implementation project is in the UK. The local implementation projects in the United Arab Emirates, Singapore, Mexico and Canada. In such complex project structures, it is important that organizations make strategic IT investments by improving enterprise architecture in order to ensure that IT infrastructure is integrated and standardized to be effectively used in GSD.

Our results show that organizational structure plays an important role in the success of GSD project management.

Table V shows that more than half of the articles have cited "project managers' skills" as a project management success factor in GSD. A highly skilled and experienced project manager is essential for monitoring and controlling projects [15]. GSD is risk-prone and hence requires special set of leadership and decision making skills by the project manager or the project management team on the whole. Project management team's prior experience in handling GSD projects plays a prominent and imminent role for the project success.

The 'communication' (54%) is the third frequently mentioned success factor in our study. Since the development sites are spread across geographical boundaries, communication between different sites is very important. Different studies have described the impact of communication on GSD projects: Tsuji et al. [16] concluded that communication capabilities have a significant impact on the results of GSD projects; Ericksen and Ranganathan [17] described the case of one offshore software development outsourcing project, which completely failed due to the lack of adequate communications. Communication is generally of two types, i.e., synchronous and asynchronous. By synchronous communication, we mean face-to-face meetings and discussion with team members and client. As GSD is different from a collocated development due to the geographically distributed teams, communicating face-to-face is not possible unless team members travel between development sites. Lack of face to face meetings can impact on other project management challenges like misunderstanding of requirements, lack of team awareness and lack of trust in GSD [7,18]. Hence, GSD relies on other synchronous and asynchronous communication channels, such as e-mail, voice mail, instant messenger, teleconferencing, and web conferencing to promote communication.

TABLE V. LIST OF FACTORS

Factors	**Freq. (n=118)**	**%**
Organizational structure	73	62
Project managers' skills	69	58
Communication	64	54
Requirement specification	48	41
Cultural awareness	47	40
Trust building	41	35
Collaboration	40	34
Work dynamics	38	32
Shared Knowledge	34	29
Team commitment and structure	31	26
Time-zone difference awareness	27	23
Cost assessment	23	19
Roles and responsibilities	17	14
Shared goals	14	12
Customer awareness	11	9
Training	10	8
Time to delivery	9	8
Incremental cycles	7	6

Requirements specification factor has been mentioned by 41% of the articles. We consider requirements specification important because it is an official statement of the system requirements for customers, end-users, software-developers, system test engineers and system maintenance staff. Indeed, the requirements document can act as a contract between customers and developers. The key to requirements specification is to present the idea of a shared understanding. In other words, all parties should be able to read this document as if it is their own.

In our study, 40% of the articles have mentioned 'cultural awareness' as one of the project management success factors in GSD projects. This is due to the fact that in a global software environment the development sites are spread across the globe, which invites cultural challenges for the project manager to handle. Due to cultural differences it is always difficult for both the client and vendor organizations to communicate with each other as the native language will, generally, not be the same [19]. Messages can be misinterpreted by different cultures, which can cause confusion and misunderstandings between different teams [20]. Hence, we can deduce that cultural awareness can improve other project management success factors, such as communication and trust, etc.

One of the project management success factors in GSD projects is creating confidence and trust among different teams [3,5]. This has been depicted in our SLR study where 35% of the articles have mentioned this as a project management success factor in GSD projects. In general, researchers agree that trust refers to an aspect of a relationship between client and vendor in which parties are willing to establish a relationship that will result in a positive desired outcome. It is always difficult to create such a relationship unless one is fully familiar with all members of the globally distributed team.

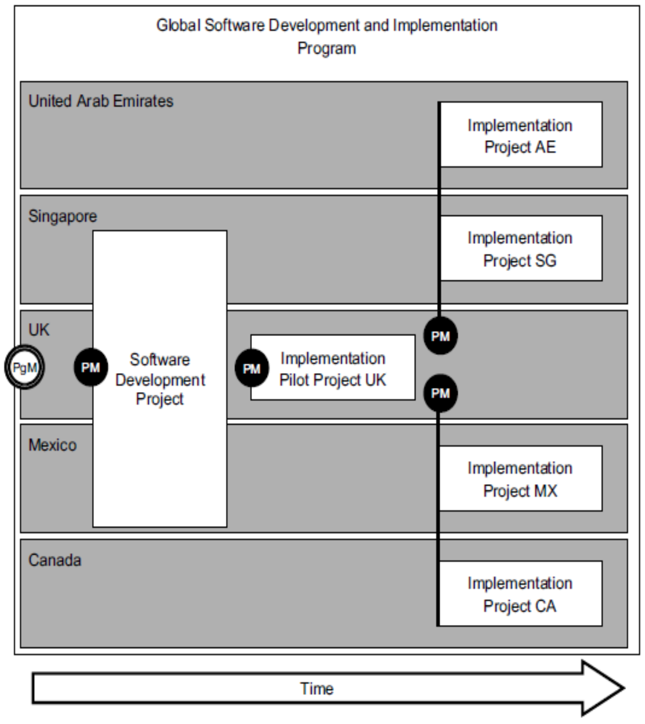

Global Software Development and Implementation Program
United Arab Emirates
Implementation Project AE
Singapore
Implementation Project SG
UK
PgM
PM
Software Development Project
PM
Implementation Pilot Project UK
PM
PM
Mexico
Implementation Project MX
Canada
Implementation Project CA
Time

Figure 1 Local program of global projects [14]

Collaboration has been mentioned in about 34% of the articles. The main reason for this factor is the difference in time zone between different development sites [21]. The other reasons for this factor include geographical and socio-cultural distance [22]. This factor can have impact on other project management factors in GSD projects such as change management activities, trust and conflict management.

Other challenges are less frequently mentioned as shown in Table V. In work dynamics, there is a continuous change and progression in the work activities, which may lead to a better outcome. This success factor has been mentioned in 32% of the articles. About 29% of the articles have described shared knowledge as a project management success factor in GSD projects. This is a very important factor as knowledge sharing is essential for any kind of project transition [20]. Since staff turnover is generally high in offshore locations, improper knowledge sharing can lead to project management issues, such as poor quality of software artifacts and documents and lack of team awareness.

V. CONCLUSION AND FUTURE WORK

The GSD is a modern software engineering paradigm. Many companies are adopting GSD to reduce software development cost and improve quality. Vendor organizations are struggling to compete internationally in attracting software development projects. Due to the increasing trend of GSD we are interested in discovering project management challenges in GSD projects. In this paper, we identified a list of success factors for project management in GSD. Among the 18 identified factors, we found that organizational structure, project manager's skills and communication and are the most common success factors.

The second phase of this research involves conducting an empirical study with the software industry to validate our findings and to provide a set of best practices, which can be used to implement these factors. The overarching objective of this research work is to develop a global project management framework to assist software development organizations in measuring and improving their project management readiness prior to starting any GSD activities**.**

VI. ACKNOWLEDGEMENT

The authors would like to acknowledge the support provided by the Deanship of Scientific Research at King Fahd University of Petroleum and Minerals, Saudi Arabia, under Research Grant 11-INF2152-04.

REFERENCES

[1] S. U. Khan, M. Niazi, and R. Ahmad, "Factors influencing clients in the selection of offshore software outsourcing vendors: an exploratory study using a systematic literature review," Journal of Systems and Software, vol. 84, no. 4, (2011), pp. 686-699.

[2] A. A. Bush, A. Tiwana, and H. Tsuji, "An Empirical Investigation of the Drivers of Software Outsourcing Decisions in Japanese Organizations," Information and Software Technology Journal, vol. 50, no. 6, (2008), pp. 499-510.

[3] M. Ali-Babar, J. Verner, and P. Nguyen, "Establishing and maintaining trust in software outsourcing relationships: An empirical investigation," The Journal of Systems and Software, vol. 80, no. 9, (2007), pp. 1438–1449.

[4] L. McLaughlin, "An eye on India: Outsourcing debate continues.," IEEE Software, vol. 20, no. 3, (2003), pp. 114-117.

[5] S. U. Khan, M. Niazi, and A. Rashid, "Barriers in the selection of offshore software development outsourcing vendors: an exploratory study using a systematic literature review," Journal of Information and Software Technology, vol. 53, no. 7, (2011), pp. 693-706.

[6] D. Damian, L. Izquierdo, J. Singer, and I. Kwan, "Awareness in the Wild: Why Communication Breakdowns Occur," International Conference on Global Software Engineering, (2007), pp. 81-90.

[7] M. Niazi, N. Ikram, M. Bano, S. Imtiaz, and S. U. Khan, "Establishing trust in offshore software outsourcing relationships: an exploratory study using a systematic literature review," IET Software, vol. in press for publication, no. (2013), pp.

[8] A. Stetten, v., D. Beimborn, E. Kuznetsova, and B. Moos, "The Impact of Cultural Differences on IT Nearshoring Risks from a German Perspective," in Proceedings of the 43rd IEEE Hawaii International Conference on System Sciences, (2010), pp. 1-10.

[9] B. Shao, David, J.S., "The impact of offshore outsourcing on IT workers in developed countries.," Communications of the ACM, vol. 50, no. 2, (2007), pp. 89 - 94.

[10] L. Mary, and R. Joseph, "Effects of offshore outsourcing of information technology work on client project management," Strategic Outsourcing: An International Journal, vol. 2, no. 1, (2009), pp. 4-26.

[11] N. B. Moe, D. Smite, and G. K. Hanssen, "From Offshore Outsourcing to Offshore Insourcing: Three Stories," IEEE Seventh International Conference on Global Software Engineering (ICGSE), (2012), pp. 1-10.

[12] D. Smite, C. Wohlin, T. Gorscheck, and R. Feldt, "Empirical evidence in global software engineering: a systematic review," Empirical Software Engineering, vol. 15, no. 1, (2010), pp. 91-118.

[13] B. Kitchenham, and C. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering. Keele University and Durham University Joint Report - EBSE 2007-001, EBSE 2007-001 (2007).

[14] J. Binder, Global project management – communication, collaboration and management across borders. Gower.(2010).

[15] S. U. Khan, M. Niazi, and R. Ahmed, "An empirical investigation of success factors for offshore software development outsourcing vendors," IET Software, vol. 6, no. 1, (2012), pp. 1-15.

[16] H. Tsuji, A. Sakurai, K. Yoshida, A. Tiwana, and A. Bush, "Questionnaire-Based Risk Assessment Scheme for Japanese Offshore Software Outsourcing," SEAFOOD07, Springer, (2007), pp. 114-127.

[17] J. M. Ericksen, and C. Ranganathan, "Project Management Capabilities: Key to Application Development Offshore Outsourcing," IEEE 39th Hawaii International Conference on System Sciences, (2006), pp. 199b.

[18] G. Aranda, N., A. Vizcaíno, and M. Piattini, "A framework to improve communication during the requirements elicitation process in GSD projects," Requirements engineering, vol. 15, no. 4, (2010), pp. 397-417.

[19] H. Christiansen, Munkebo, "Meeting the challenge of communication in offshore software development," Software Engineering Approaches for Offshore and Outsourced Development. Lecture Notes in Computer Science, vol. 4716, no. (2007), pp. 19-26.

[20] I. Musio, "IBM Industry Practice: Challenges in Offshore Software Development from a Global Delivery Center," Software Engineering Approaches for Offshore and Outsourced Development. Lecture Notes in Business Information Processing, vol. 35, no. (2009), pp. 4-13.

[21] A. Begel, and N. Nagappan, "Global Software Development: Who Does It?," IEEE International Conference on Global Software Engineering, (2008), pp. 195,199.

[22] P. Bannerman, E. Hossain, and R. Jeffery, " Scrum Practice Mitigation of Global Software Development Coordination Challenges: A Distinctive Advantage?," 45th IEEE International Conference on System Science (HICSS), (2012), pp. 5309 - 5318.

A DSL for Multi-Scale and Autonomic Software Deployment

Raja BOUJBEL, Jean-Paul ARCANGELI
Université de Toulouse UPS - IRIT
France
Raja.Boujbel@irit.fr, Jean-Paul.Arcangeli@irit.fr

Sébastien LERICHE
Université de Toulouse ENAC
France
Sebastien.Leriche@enac.fr

***Abstract*—In this paper, we present an ongoing work which aims at defining and experimenting a Domain-Specific Language (DSL) dedicated to multi-scale and autonomic software deployment. Autonomic software deployment in open environments is an open issue. There, the topology of target hosts is not always known due either to unforeseen hardware failures or limitations (network links, hosts, etc.) or to device arrival and disappearance. In a previous work, we proposed to describe deployment constraints using a DSL and then to satisfy them using a middleware for autonomic deployment, rather than classically building and executing a deployment plan. As deployment of multi-scale distributed systems demands the expression of specific constraints related to dimensions and scales, it is necessary to think over and define a new Domain-Specific Language. In this paper, we propose a new DSL designed to support the expression of constraints and properties related to multi-scale and autonomic software deployment.**

***Index Terms*—Deployment, Multi-Scale, DSL, Component-Based Software System**

I. Introduction

Pervasive computing, on the one hand, and cloud computing, on the other hand, are central topics in several recent research studies. Contributions in both domains have reached a good level of maturity. Nowadays, new research works have identified the need to make pervasive and cloud computing systems collaborate, so to build systems which are distributed over several scales, called "multi-scale" systems.

The INCOME project [1] aims at designing software solutions for multi-scale context management, not only in ambient networks but also in the Internet of Things and clouds, able to operate at different scales and to deal with the passage from a scale to another one. Context management is a complex service in charge of the gathering, the management (processing and filtering), and the presentation of context data to applications, which realization is distributed on the different devices which compose the system. So, context managers are open multi-scale applications which must be deployed, *i.e.*, made and kept available for use, in a situation of mobility and variability of the quality of the resources. In this project, our work focuses on software deployment and our goal is to develop a framework for supporting the deployment of multi-scale applications such as context managers. Deployment strategies should take into account the multi-scale aspects like geography, network, device, and user, as well as non functional properties such as efficiency and privacy. In multi-scale systems, decentralization, autonomy and adaptiveness are essential features.

In this paper, we present an ongoing work which aims at defining and experimenting a Domain-Specific Language (DSL) dedicated to multi-scale and autonomic software deployment.

The paper is structured as follows. Section II introduces the two main aspects of our working context: multi-scale distributed systems and software deployment. Section III provides an example of deployment of a multi-scale software system, analyses the requirements, and proposes to use a DSL to support autonomic deployment. Section IV discusses related work on DSL for software deployment. Our DSL is presented in Section V using the example presented in Section III. Section VI concludes and discusses some perspectives.

II. Context Overview

This section introduces the novel concept of multi-scale system and provides an overview of software deployment.

A. Multi-scale distributed systems

The term "multi-scale system" is present in several recent research papers [2], [3], [4]: in these works, authors consider to make collaborate very small systems (objects from the Internet of Things paradigm as, for example, swarms of tiny sensors with very low computing capabilities) with very big systems (such as those found in cloud computing). They agree that new issues arise, mainly those related to huge heterogeneity.

In [5], authors argue that the multi-scale nature of a distributed system should be analyzed independently in several specific dimensions such as geography, network, device, data, user, etc. Thus, a distributed system can be described as multi-scale when, for at least one dimension, the elements of its projection onto this dimension are associated with different scales. Fig. 1, extracted from [6], shows an example of scales in the "Device processing power" dimension.

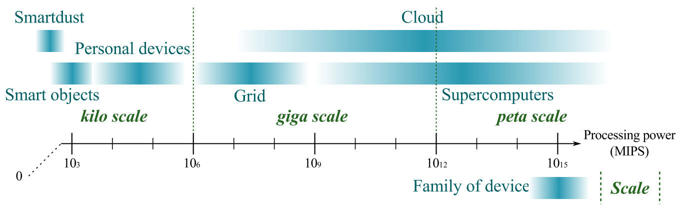


Fig. 1: Scales in the "Device processing power" dimension

However, the concept of "multi-scale system" is not actually mature. The construction of future multi-scale distributed systems will necessitate a new kind of languages, middleware and patterns, allowing to take in consideration the multi-scale aspects of the systems.

B. Software deployment

Software deployment is a post-production process which consists in making software available for use and then keeping it operational. It is a complex process that includes a number of inter-related activities such as installation of the software into its environment (transfer and configuration), activation, update, reconfiguration, deactivation and deinstallation [7]. Fig. 2 represents the sequence of the activities. Software release and software retire are carried out on the "producer site", while the other activities are carried out on the "deployment site", some of them at runtime.

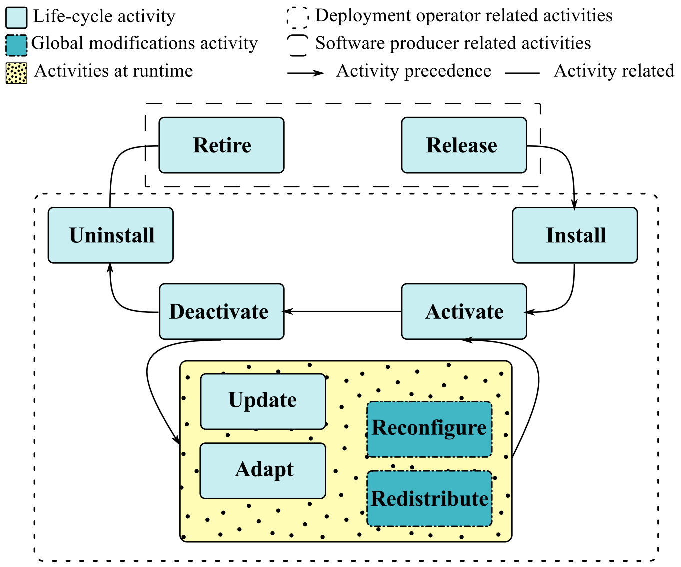


Fig. 2: Software deployment life cycle

Deployment design is handled by an engineer called "deployment designer". He has to gather information not only about the software system to deploy and the properties of each of its components but also about the distributed organization of the software at runtime. Designing deployment may consist in expressing properties (commands, requirements, etc.) and constraints. For instance, the deployment designer may express that a particular software component should be installed on some specific devices or on any device, even on incoming ones in case of dynamic systems, while satisfying a set of constraints. As a concrete example, consider a software component C which should be deployed on each smartphone which runs Android, has the GPS function active, and is connected by WiFi.

A deployment plan is a mapping between a software system and the deployment domain, increased by data for configuration. The deployment domain is a distributed set of machines which host the software system and provides resources to it. The ultimate purpose of deployment design is to produce a deployment plan which complies with the expressed properties and constraints. Usually, this task is undertaken by a human actor.

At runtime, software must be deployed on the domain according to the deployment plan, this task being possibly undertaken or controlled by an operator called "deployment operator". Automatization of deployment aims at avoiding (or limiting) human handling in the management of deployment.

Fig. 3 shows the timeline of deployment.

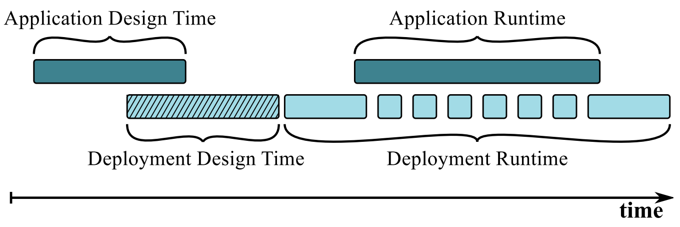


Fig. 3: Software deployment timeline

III. Deployment of multi-scale software systems

In this work, we focus on the design phase of the deployment process, and precisely on the ways for a deployment designer to express deployment properties and constraints.

Here is an example, in order to illustrate our aim. Let's consider a software system made of different components, each of them having specific individual runtime requirements (memory, OS, etc.). The deployment designer may want to express not only these requirements, but also some other ones related to the distribution of the components. For instance, the deployment designer may want that (C1...C5 are software components):

- a resource-consuming component C1 runs on a cloud,
- C2 runs on several machines in a given geographical area, *e.g.*, a city,
- C3 runs on the same device than C1,
- C4 runs on any smartphone of the domain,
- C5 runs on the same network than C4,
- C4 runs on any new smartphone entering in the domain at runtime.

Moreover, some components may have constraints to run properly, such as:

- C1 requires that the component C0 is installed and activated locally,
- C2 must run on a Linux OS and an Arduino (single-board microcontroller) must be connected to the hosting device,
- C3 requires 40M of free RAM at activation time (Constr1),
- C5 requires a 100G hard drive (Constr2).

Fig. 4 illustrates such an example.

This section analyses the problem of software deployment of multi-scale systems from the design point of view, and then motivates the use of a Domain-Specific Language which supports the expression of multi-scale deployment properties and constraints.

A. Analysis

Software deployment in large-scale and open distributed systems (such as ubiquitous, mobile or peer-to-peer systems) is still an open issue [8]. There, existing tools for software deployment are reaching their limits: they use techniques

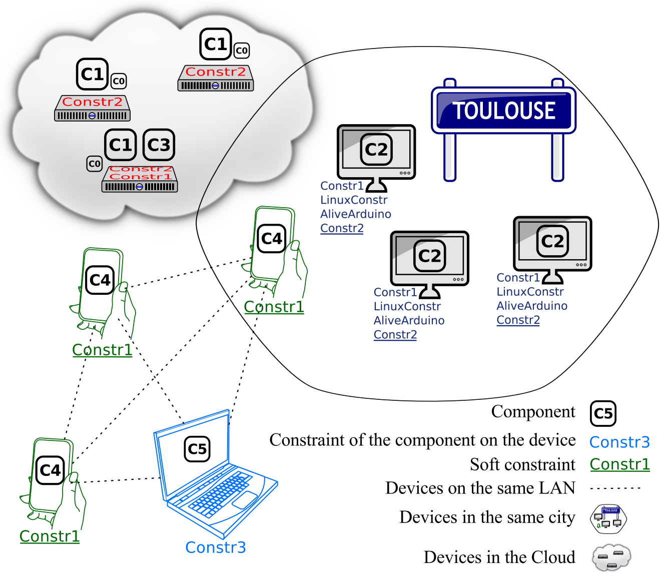


Fig. 4: Example of multi-scale deployment

that do not suit the complexity of the issues encountered in such infrastructures. Indeed, they are only valid within fixed network topology and do not take into account neither host and network variations of quality of service nor failures of machines or links which are typical of these environments.

In addition, users of the deployment tools are required to manage manually the deployment activities, which needs a significant human involvement, possibly out of reach of concerned end-users (for example, in case of personal devices like smartphones): for large distributed component-based applications with many constraints and requirements, it is too hard and complicated to accomplish the deployment process manually. Consequently, there is a need for new infrastructures and techniques that automate the deployment process and allow a dynamic reconfiguration of software systems with few or without human intervention.

Additionally, in our opinion, decentralization, openness and dynamics (mobility, variations of resources availability and quality, disconnections, failures) are in favour of autonomy: the autonomic computing approach [9], where the system self-manages some properties (self-configuration, self-healing), may support solutions which satisfy the requirements of distributed multi-scale software systems deployment. This idea lead us to “autonomic software deployment” [8].

B. Our approach

Instead of directly expressing a statically defined deployment plan, we propose to express deployment constraints and properties from which the deployment plan can be computed. In this paper, we focus on the expression of the constraints and properties, not on the construction of the plan. For this last point, our idea is use a constraint solver, supplying it with an up-to-date description of a domain (available hosts and their properties).

So, in order to build the plan, and moreover to allow management of deployment at runtime, data about the domain must be collected. Thus, a system of probes should run and collect data ranging from the domain properties such as free RAM to more abstract ones related to multi-scale (dimensions and scales). Relations between probes and properties can be made explicit at the same level as the deployment properties and constraints in order to allow the specification of the system of probes at the deployment design time.

C. Towards a DSL for autonomic software deployment of multi-scale systems

In this ongoing work, our aim is to provide a solution for the expression of the deployment design, concerning in particular the dimensions and other significant properties of multi-scale software systems.

Deployment is a specific operation on software. Its design requires particular skills. Thus, we think that the deployment designer could benefit from a dedicated language when stating the properties and constraints. So, we propose a DSL dedicated to the description of deployment constraints and properties. DSLs present several advantages: they use idioms and abstractions of the targeted domain, so they can be used by domain experts; they are light, so easy to maintain, portable, and reusable; they are most often well documented, coherent and reliable, and optimized for the targeted domain [10], [11], [12].

IV. Related work on DSL for software deployment

Existing deployment platforms propose several formalisms to express deployment constraints, software dependencies, and hardware preferences of software to deploy. Usually, the formalisms include architecture description languages (ADL), deployment descriptors (like XML descriptor deployment), and dedicated languages (DSL). In this section, we overview some works on software deployment that propose the use of a DSL.

Dearle *et al.* [13], [14] present a framework for autonomic management of deployment and configuration of distributed applications. To facilitate the work of the deployment designer, they define a DSL, Deladas. Using it, a set of available resources and a set constraints are specified. These definitions permit to generate an applicable deployment plan. The constraint-based approach avoids the deployment designer specifying precisely the location of each component, and then rewriting all the plan in case of problems with a resource. Deladas does not allow to express multi-scale properties and constraints. Openness is neither taken into account, the set of hosts is statically defined in a file by the deployment manager. Deployment is still autonomic: at runtime, when the deployment middleware detects a constraint violation (dependencies between components), it tries to solve it by a local adaptation. The new deployment plan is computed by a centralized management component called MADME.

Matougui *et al.* [8] present a middleware framework designed to reduce the human cost for setting up software deployment and to deal with failure-prone and change-prone environments. This is achieved by the use of a high-level constraint-based language and an autonomic agent-based system for establishing and maintaining software deployment. In the DSL (called j-ASD), some expressions dedicated to deal with autonomic issues are proposed. But they target large-scale or dynamic environments such as grids or P2P systems, only within the same network scale.

Sledviewsky *et al.* [15] present an approach that incorporate DSL for software development and deployment on the cloud. Firstly, the developer defines a DSL in order to describe a model of the application with it. Secondly, this model is translated into specific code and automatically deployed onto the Cloud. This approach is specific to deployment on the cloud. It highlights the need to facilitate the work of the deployment designer, and that using DSL is a solution for that.

V. Proposition of a DSL

In this section, we describe by means of an example our proposition of a DSL dedicated to the autonomic deployment of multi-scale distributed systems. Tokens and keywords are presented further and the grammar is defined in EBNF syntax. The grammar is available at http://www.irit.fr/~Raja.Boujbel/ebnf-jasd.html.

A. Example

We give below (Listing 1) a full example of code for the deployment of the multi-scale distributed software system presented in Section III. Then, we use this code to present and explain the main elements of the language.

```
Include "base.jasd"
//base.jasd defines some probes
//like OS, RAM, CPU, Network, and HD

Component C0 {
    Version 1
    URL "http://test.fr/plopC0.jar"
}

Component C1 {
    Version 1
    URL "http://test.fr/plopC1.jar"
    Require C0
    DeploymentInterface fr.enac.plop.DIimpl
}

Probe Arduino {
    ProbeInterface fr.irit.arduino.DIimpl
    URL "http://irit.fr/INCOME/arduinoProbe.jar"
}

Constraint AliveArduino {
    Arduino Exist, Alive
}

Constraint LinuxCstr {
    OS.Name = "Linux"            //OS probe
}

Constraint Constr1 {
    RAM.FreeSpace >= 40          //RAM probe
}

Constraint Constr2 {
    CPU.Load < 80                //CPU probe
    Network.BandWith > 1024      //Network probe
}

Constraint Constr3 {
    HD.size > 100                //HD probe
}

Component C2 {
    Version 1
    URL "http://test.fr/plopC2.jar"
    DeploymentInterface fr.enac.plop.DIimpl
    Constraint Constr1, LinuxCstr, AliveArduino
    Soft Constraint Constr2
}

Component C3 {
    Version 1
    URL "http://test.fr/plopC3.jar"
    DeploymentInterface fr.enac.plop.DIimpl
    Soft Constraint Constr1
}

Component C4 {
    Version 1
    URL "http://test.fr/plopC4.jar"
    DeploymentInterface fr.enac.plop.DIimpl
    Soft Constraint Constr1, Constr2
}

Component C5 {
    Version 2
    URL "http://irit.fr/plopC5.jar"
    Constraint Constr3
}

MultiScaleProbe Geography {
  MultiScaleProbeInterface
    eu.telecom-sudparis.GeographyProbeImpl
  URL "http://it-sudparis.eu/INCOME/GeoProbe.jar"
}

//other MultiScale probes are described
//the same way
//{...}

Deployment {
    AllHosts LinuxCstr

    C1 @ Constr2, Device.Cloud
    C2 @ 2..4 Geography.City("Toulouse")
    C3 @ SameValue Device(C1)
    C4 @ All Device.SmartPhone
    C5 @ SameValue Network.MAN(C4)
}
```

Listing 1: Example of code for the deployment of the multi-scale distributed software system

B. Elements of the language

1) Component: The keyword `Component` defines a component. The `Version` field is useful for the update activity. The `URL` field specifies the address where the component is reachable for download. The `DeploymentInterface` field specifies the interface of the component, necessary for the interactions with the deployment system: the latter must interact with the component, for configuring and starting it, for managing it at runtime, and for stopping it. The **`Require`** field lists required components: at installation time of the component, if the required component is not installed, the deployment system must install it on the device. The **`Constraint`** field lists hardware and software constraints of the component. By default, these constraints are hard, *i.e.*, they must be satisfied both when generating the deployment plan and at runtime (so, the deployment system must check that there is no constraint violation). For the keyword **`Soft`**, see *6)*.

2) Probe: The keyword **`Probe`** defines a probe. A probe has two mandatory fields. The first one, the **`Probe-Interface`**, specifies the interface of the probe. This interface is needed for interactions with the deployment system for information retrieval. The second one, the **`URL`**, specifies the address where the probe is reachable for download.

3) Constraint: The keyword **`Constraint`** defines a constraint on a component. It has one kind of field, a probe value test. There can be several tests in a **`Constraint`**, like in `Constr2` (line 35). A probe value test is composed by two or three parts. If the constraint is related to the existence or the liveliness of a hardware or a software component, the probe value test is composed by the probe name and keywords **`Exists`** or **`Alive`**. These keywords are defined for any probe interface. For example, at line 23, the used probe is `Arduino`, and the constraint uses default methods **`Exists`** and **`Alive`**. If the constraint is about a value, the probe value is composed by the probe name, a method call, a comparator, and a value. There, the method is probe specific, and defined in the probe interface For example, at line 27, the used probe is `OS`, the information method used is `Name`, and its value is compared to the string `"Linux"`.

4) Multi-scale Probe: The keyword **`MultiScaleProbe`** defines a multi-scale probe, useful for the deployment. Like **`Probe`**, it has only two fields. The first one, **`MultiScale-ProbeInterface`**, specifies the interface of the probe. The second one, **`URL`** specifies the address where the implementation of the probe is reachable for download. In our current solution, scales are defined in the implementation of probes, and the probes allows to identify the scale of a given device.

5) Deployment: The keyword **`Deployment`** defines the deployment properties and constraints. The keyword **`AllHosts`** allows to specify and delimit the deployment domain: line 83 expresses that the deployment covers all hosts which satisfy the constraint `LinuxCstr`. The operator `@` allows to specify deployment constraints specific to a component. These constraints can take several forms: the device hosting the component `C1` must satisfy `Constr2` and be on the scale `Cloud` on the dimension `Device` (line 85); the component `C2` must be deployed on `2` to `4` devices, in the city `Toulouse` (line 86); the component `C4` must be deployed on all devices of the dimension `Device.Smartphone`, *i.e.*, on all smartphones of the domain (line 88). The keyword **`SameValue`** expresses that the component must be in the same dimension or scale as a referred one: the component `C3` (line 87) must be deployed on one device (implicit) which has the same value in the dimension `Device` as the device hosting `C1` (in other words, `C3` should be deployed on the same device as `C1`); the component `C5` must be deployed on a device which is situated in the same medium area network (MAN) as the device hosting `C4` (line 89).

6) Dynamics and openness: Some constructions of the DSL are particularly well-adapted for the expression of properties related to dynamics and openness. By default, the constraints should be satisfied during the entire application runtime, and so must be checked dynamically. The keyword **`Soft`** is used to specify that a constraint should be satisfied initially by the generated deployment plan, but maybe not satisfied at runtime. When specifying the **`Deployment`**, the keyword **`All`** allows to specify that a component should be deployed on a subdomain which satisfies (even dynamically) a property or a constraint. In the example, the component `C4` should be deployed on every smartphone of the domain, including those which enter in the domain at runtime; so, the deployment plan evolves dynamically depending on entering and leaving devices.

The file must have at least one definition of a component and one expression of the deployment. Other fields are optional. As the code can be split in several files, the keyword **`Include`** permits to include other files (line 1).

VI. Conclusion and future work

In this paper, we present the first version of a DSL for multi-scale and autonomic deployment, and explain the various elements of the language by means of an example. This DSL allows to express the deployment constraints of a multi-scale software system and its components. These constraints drive the computation of the deployment plan, and are used by the autonomic deployment system do detect (and possibly repair) any constraint violation at the application runtime.

Another part of our work concerns the realization of this autonomic deployment system. We are designing it as a middleware, on the same basis than first experiments described in our previous work [8]. This middleware will enable deployment in multi-scale environments. It will provide the probes needed to gather informations about the hosts.

We believe that a DSL is the best way for a deployment designer to describe deployment constraints. A DSL has much more expressiveness than any Markup Language (such as XML), and is more efficient since the deployment designer expresses (and read) directly concepts of its field of expertise. Moreover, modern tools for making DSL allows their designers to integrate several level of validation (not only syntactic but also semantic).

Presently, the DSL targets the installation and activation activities. Other activities and features, as constraint infringement at application runtime, are hard coded in the deployment manager system. In the future, we can move some of them at the DSL level, to increase expressiveness and flexibility when designing deployment. For example, we can add in the grammar the keyword **`on-deinstall`** or **`on-update`** to define actions to perform when deinstalling or updating a component.

Focusing on multi-scale systems, we do need a sound and extensible vocabulary to describe the dimensions and their scales. In the INCOME project, another ongoing work aims at defining an ontology for multi-scale distributed systems. We plan to integrate these concepts in our DSL.

Besides, we are currently working on a toolchain for our DSL. Using Xtext and Xtend frameworks [16], we have realized an Eclipse plugin for the edition of the DSL that makes it multi-platform compliant and easy-to-use for a deployment designer. The DSL and the Eclipse plugin are part of the deliverables of the INCOME project.

ACKNOWLEDGMENTS

This work is part of the French National Research Agency (ANR) project INCOME [17] (ANR-11-INFR-009, 2012-2015). The authors thank all the members of the project that contributed directly or indirectly to this paper.

REFERENCES

[1] J.-P. Arcangeli, A. Bouzeghoub, V. Camps, C. M.-F. Canut, S. Chabridon, D. Conan, T. Desprats, R. Laborde, E. Lavinal, S. Leriche, H. Maurel, A. Péninou, C. Taconet, and P. Zaraté, "INCOME - Multi-scale context management for the internet of things," in AmI, ser. Lecture Notes in Computer Science, F. Paternò, B. E. R. d. Ruyter, P. Markopoulos, C. Santoro, E. v. Loenen, and K. Luyten, Eds., vol. 7683. Springer, 2012, pp. 338–347, doi:10.1007/978-3-642-34898-3.

[2] G. Blair and P. Grace, "Emergent middleware: Tackling the interoperability problem," Internet Computing, IEEE, vol. 16, no. 1, pp. 78–82, Jan.-Feb. 2012, doi:10.1109/MIC.2012.7.

[3] M. Kessis, C. Roncancio, and A. Lefebvre, "DASIMA: A flexible management middleware in multi-scale contexts," in Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on, April 2009, pp. 1390–1396, doi:10.1109/ITNG.2009.338.

[4] M. van Steen, G. Pierre, and S. Voulgaris, "Challenges in very large distributed systems," Journal of Internet Services and Applications, vol. 3, no. 1, pp. 59–66, 2012, doi:10.1007/s13174-011-0043-x.

[5] S. Rottenberg, S. Leriche, C. Lecocq, and C. Taconet, "Vers une définition d'un système réparti multi-échelle," in Journées francophones Mobilité et Ubiquité (UBIMOB). Cépaduès Editions, 2012, In French.

[6] S. Rotteneberg, S. Leriche, C. Taconet, C. Lecocq, and T. Desprats, "From Smartdust to Cloud: The emergence of multiscale distributed systems," 2013, Unpublished.

[7] A. Carzaniga, A. Fuggetta, R. S. Hall, D. Heimbigner, A. Van Der Hoek, and A. L. Wolf, "A characterization framework for software deployment technologies," DTIC Document, Tech. Rep., 1998.

[8] M. E. A. Matougui and S. Leriche, "A middleware architecture for autonomic software deployment," in ICSNC '12 : The Seventh International Conference on Systems and Networks Communications. Lisbon, Portugal: XPS, 2012, pp. 13–20, 12619 12619 . [Online]. Available: http://hal.archives-ouvertes.fr/hal-00755352

[9] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," Computer, vol. 36, no. 1, pp. 41–50, 2003, doi:10.1109/MC.2003.1160055.

[10] A. Van Deursen, P. Klint, and J. Visser, "Domain-specific languages: An annotated bibliography," ACM Sigplan Notices, vol. 35, no. 6, pp. 26–36, 2000.

[11] M. Strembeck and U. Zdun, "An approach for the systematic development of domain-specific languages," Software: Practice and Experience, vol. 39, no. 15, pp. 1253–1292, 2009, doi:10.1002/spe.936.

[12] J.-P. Tolvanen and S. Kelly, "Integrating models with domain-specific modeling languages," in Proceedings of the 10th Workshop on Domain-Specific Modeling, ser. DSM '10. New York, NY, USA: ACM, 2010, pp. 10–1, doi:10.1145/2060329.2060354.

[13] A. Dearle, G. N. C. Kirby, and A. J. McCarthy, "A framework for constraint-based deployment and autonomic management of distributed applications," in ICAC, ser. 1st International Conference on Autonomic Computing (ICAC 2004), New York, NY, USA. IEEE Computer Society, May 2004, pp. 300–301, doi:10.1109/ICAC.2004.3.

[14] A. Dearle, G. N. C. Kirby, and A. McCarthy, "A middleware framework for constraint-based deployment and autonomic management of distributed applications," CoRR, vol. abs/1006.4733, 2010.

[15] K. Sledziewski, B. Bordbar, and R. Anane, "A DSL-based approach to software development and deployment on cloud," in AINA, ser. 24th IEEE International Conference on Advanced Information Networking and Applications, AINA 2010, Perth, Australia, 20-13. IEEE Computer Society, April 2010, pp. 414–421, doi:10.1109/AINA.2010.81.

[16] M. Eysholdt and H. Behrens, "Xtext: implement your language faster than the quick and dirty way," in SPLASH/OOPSLA Companion, ser. Companion to the 25th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, SPLASH/OOPSLA 2010, Reno/Tahoe, Nevada, USA, W. R. Cook, S. Clarke, and M. C. Rinard, Eds. ACM, October 2010, pp. 307–309, doi:10.1145/1869542.1869625.

[17] "INCOME," http://anr-income.fr, February 2012, last access 2013.

Characterization of Techniques and Tools of Visualization Applied to Software Comprehension

A Systematic Mapping

Marllos Paiva Prado
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: marllosprado@inf.ufg.br

Auri Marcelo Rizzo Vincenzi
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: auri@inf.ufg.br

Fabrízzio Alphonsus A. de M. N. Soares
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: fabrizzio@inf.ufg.br

Felipe César
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: kamuie4ever@hotmail.com

Guilherme Pereira de Paula
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: guilhermepaula@inf.ufg.br

Hugo Alexandre Dantas do Nascimento
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: hadn@inf.ufg.br

João Carlos Silva
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: jcs@inf.ufg.br

Juliano Lopes de Oliveira
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: juliano@inf.ufg.br

Lucas Carvalho Lima
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail: lucaslima@inf.ufg.br

Thiago Fernandes
Instituto de Informática
Universidade Federal de Goiás, UFG
Goiânia-GO, Brazil
e-mail:hidekownz4@hotmail.com

***Abstract*—This study aimed to make a characterization about how information visualization has been applied on software comprehension with relation to techniques and tools proposed by literature. Systematic Mapping was adopted as the method to guide the investigation process. The findings, although not being definitive, points direction over important questions such as what kind of artifacts and life-cycle phase have been more considered. It was also investigated how these studies have been evaluated, how they evolved in the main digital libraries over the last decade, and what points deserves further attention, through new research.**

Keywords – Systematic; Mapping; Visualization; Sofware Comprehension

I. Introduction

Information Visualization applied to Computer Science is committed to the visual representation of abstract data handled by computer and to interaction as a way of magnifying the cognition [1]. Data to be modeled does not necessarily need to have an intrinsic geometry shape previously associated [2]. This research field aims to develop and apply visual models to explore the human cognitive abilities of recognizing and deriving information from graphics of important data and their relationships [3].

One of the best ways to minimize the complexity of creating/maintaining a system is to simplify its understanding. At this point, an adequate visualization of the artifacts and information generated in the development process can be a favorable factor. The reason is that human vision has advantageous attributes such as the power to capture a high amount of information in parallel, considering a short period of time, and also the capacity to focus attention in a object's fragment of interest while not losing the attention of what happens at its surrounding [4]. Therefore, the characterization of manner and goals in which these proposals have been applied is a way of better knowing the existing solutions and what research opportunities are opened in this field.

Experimental Software Engineering provides guidance in a well standard and organized conduction of software engineering experiments. As mentioned by Wohlin et al. [5], experimental studies let one object of interest to be evaluated by different people and environments. The more an experiment is repeated, considering different contexts, the more information is obtained about the object of study, so that the results are more significant.

Two kinds of experimental studies have been disseminated in the research community: systematic review [6] and systematic mapping [7]. Both are secondary experimental studies and differ subtly. While the systematic mapping focuses at mapping the related research to some question of interest through a detailed categorization of primary related study, the systematic review is more restrictive, as a way to identify, evaluate and compare qualitatively all the relevant research to a research question.

The next five sections of this paper present the definition, execution and results obtained from a systematic mapping to characterize the visualization applied to the software comprehension. The adopted process on this work followed the definitions of Petersen, Feldt, Mujtaba and Mattsson [7] and Kitchenham [8] once these works target the definition of parameters for experimental studies applied to software engineering themes. Paper organization follows the sequence order adopted at StArt tool [9]. Section II describes the method and the parameters adopted in the planning and definition of a study. Description of the intermediate steps and results gotten from the investigation conduction as stated in the pre-established plan is defined and related at Section III. Section IV presents results reached after extraction step in the systematic review and gather relevant information observed from the analysis. Section V raises question about threats of validity concerning this study. Section VI concludes this paper and discusses future research.

II. PLANNING

Systematic Mapping planning is composed by goals definition, protocol, research strategy and inclusion/exclusion criteria to be adopted. Protocol consists of research question definition, population, intervention, control, results and application. These parameters were specified as follows.

A. Research Goals

This research consists of characterizing how visualization has been used at software comprehension, through the identification of papers which discuss tools and techniques applied to software comprehension.

B. Main Questions

Based on the defined goal, the following research questions were formalized:

- R.Q.1. How publications about visualization tools and techniques, applied to software comprehension, have been evolved in the main digital libraries?
- R.Q.2. How visualization tools and techniques, applied to software comprehension, have been evaluated?
- R.Q.3 What is the profile of visualization tools and techniques, applied to software comprehension, considering the artifacts represented, life-cycle phase and training?

C. Population

The population considered was composed by researchers and developers, who use/propose information visualization at software comprehension and publish them at indexed electronic databases.

D. Intervention

The observed characteristic was the application of tools and visualization techniques to the software comprehension. Characteristic Observation was made from software engineering researchers' point of view.

E. Control

A total of four relevant papers [10] [11] [12] [13] were previously set by experts to be used as the control for the search string. The search result shall be considered adequate in case of returning all these papers in the considered databases.

F. Results

The expected result in the end of the systematic mapping is the characterization of relevant information about visualization tools/technique application at software comprehension.

G. Application

Systematic mapping results should collaborate to a better understanding of how visualization has been applied until the moment at software comprehension, so it lets the identification of weakness and opportunities in this research field.

H. Research Strategy and Search String

Most bibliographic studies, including systematic mapping and reviews, are made through an automatic search over digital libraries and using a pre-defined search string. This string is a set of combined key-words which reflects the search to be made, organized in a way to guarantee that the returned results of the search be closer as possible from the scope. According to the formulated research questions, the following search string was used:

- *Q0: ("technique*"OR "tool*") AND ("visuali*ation") AND ("comprehension"OR "understanding") AND ("software"OR "program")*

Following the recommendations from Kitchenham [8], besides key-words, specific synonymies and spelling variations to our research question were considered. This approach was employed to increase the number of returned papers and to avoid that important papers to our research question were neglected because of synonymy word, e.g., "understanding" instead of "comprehension".

I. Tools and Instrumentation

The digital libraries adopted to do the search in this systematic mapping were: "IEEE Xplore", "ACM" and "Science Direct". Additionally, it was applied a manual search to complement it, that is, a search made at Google website, using some terms of the search string and applying a case by case analysis to decide if the paper would be pertinent or not.

During the execution step, all returned papers after applying the search string in the digital libraries should be first considered as relevant and added to the StArt tool [9], for classification aftermost. The same has occurred in the manual search. This approach was followed because StArt tool divides the execution process in three phases:

1. **Study Identification**: Phase to when the automatic and manual returned papers are added to the tool. Information such as name, title, author, abstract, publishing source and year of the paper are organized to make reviser work easier, so the study conduction can become a less costly work.
2. **Selection**: Phase when an initial search is made. At this step, revisers are responsible for the abstract reading and to approve or reject the paper to the next step. This step is of great relevance, since it is the first phase which eliminates from the whole paper set, those that clearly are not part of the research scope.
3. **Extraction**: Phase when the selected papers from previous step are read and decision to accept or not the article to the final step is definitive. Besides that, the tool lets pre-defined classification studies on the plan to be verified. These data are after used in the automatic generation of statistics which will serve as the bases to the final conclusions of the study.

J. Inclusion and Exclusion Criteria

The following restrictions were adopted to eliminate papers considered not relevant to the study, on selection step:

- R.1. Studies which are not written in English Language.
- R.2. Studies which were not published in conference proceedings or journals.
- R.3. Studies prior to the year 2003.

Besides restrictions described, the following inclusion criteria (I.C.) and exclusion criterion (E.C) were adopted on the selection and extraction phase, related to the papers content:

- I.C.1. Addresses the application of visualization tools in the software comprehension.
- I.C.2. Addresses the application of visualization techniques in the software comprehension.
- I.C.3. Experimental studies about visualization techniques or tools in the software comprehension.
- E.C.1. Does not address the application of visualization tools or techniques in the software comprehension.

To answer the research question it is needed to analyze data collected by study conduction. These data, in turn, are obtained observing the classification criteria. Therefore, these classification criteria should represent objective and coherent properties to the parameters intervention defined in the plan. To each criteria established, the value "not adequate" was created to classify papers which do not meet any of the pre-defined values to the referred criteria, or cannot be considered a primary study. The latter intended to avoid that properties of a specific classification criterion could count for a work whose technique has a secondary focus, for example. In this study, the following classification criteria (C.C.) were adopted:

- C.C.1 – Source: Responsible to register the study source: (i) IEEE; (ii) ACM; (iii) Science Direct.
- C.C.2 – Focus: Responsible to register the study focus. Options: (i) Tools; (ii) Technique; (iii) Study – In the case that paper reports an experimental study about tools and/or technique application in software comprehension.
- C.C.3 – Evaluation Context: Responsible to register tests and results presented on the study. Options: (i) Tested in production context; (ii) Tested in academic environment; (iii) Does not show any evaluation result; (iv) Not Adequate.
- C.C.4 – Analysis Criterion: Responsible to register the way tool/technique generates the visualization. Options: (i) Static Analysis; (ii) Dynamic Analysis; (iii) Both; (iv) Not Adequate.
- C.C.5 – Object of Analysis: Responsible to register the object(s) represented in the visualization. Options: (i) Source Code; (ii) Execution Tracing; (iii) Documentation; (iv) Memory; (v) UML Diagram; (vi) Graphical Interface; (vii) Communication Register; (viii) Threads; (ix) Other artifacts; (x) Not Adequate.
- C.C.6 – Representation Type: Responsible to register the type of representation used in the visualization. Options: (i) 2-D; (ii) 3-D; (iii) Both; (iv) Not Adequate.
- C.C.7 – Life Cycle Scope: Responsible to register the knowledge focus of application of the visualization tool/technique. Options (i) Requirements; (ii) Construction; (iii) Test; (iv) Quality; (v) Maintenance; (vi) Design; (vii) Multiples; (viii) Not Adequate..
- C.C.8 – Learning Aided: Responsible to register if the tool/technique used the visualization to help the learning/training of beginners. Options: (i) Yes; (ii) No; (iii) Not Adequate.
- C.C.9 – Specific Platform or Language: (i) Name of the specific platform or language of the tool. (ii) Not Adequate

- C.C.10 – Tool Name: Responsible to register the tool name.

III. MAPPING CONDUCTION

The following subsections present the results of extraction and selection phases of systematic mapping based on the defined plan. This study was realized between November 2012 and February 2013.

During the Study Identification phase, the searches were applied and returned papers were added to the StArt [9] tool for analysis. Summing the total of returned papers by the automatic and manual search, it was obtained a total of 449 papers. From this total, 116 were considered duplicated and removed automatically by the tool. So, it was left a set of 333 papers to be evaluated.

A. Selection

At selection phase, 333 papers were submitted to analysis process. This process was defined in the following way: Initially, each paper's abstract, title and key-words were read by two revisers, in a way that determination made (inclusion and exclusion criteria) would be enough safe decided. Then, each reviser gives its advising. If both voted for the paper rejection, the paper was automatically rejected. If both voted for its preliminary acceptance, the paper was accepted until the next phase, when decisions made would be final. In case of divergent opinions, i.e., one reviser voted for its exclusion and the other for its inclusion, the paper was set as accepted, to be read on the next phase. Additionally, there were cases when two or more papers of the same author and describing the same technique were found. In this case, only the most recent paper was considered.

Applying the analysis process, the 333 papers were pre-classified. From this total, 228 (68%) of the papers were rejected (eliminated) and 105 (32%) of the papers were accepted and passed to the next step.

B. Extraction

During the extraction phase, each of the 105 papers identified on the previous phase were read and classified permanently according to the inclusion and exclusion criteria. As it can be observed in Table I, from the 105 studies, 3 were considered duplicated, 27 were rejected and 75 were accepted.

TABLE I. PAPER STATUS X MAPPING PHASE

Phase	Status		
	Duplicated	*Rejected*	*Accepted*
Identification and Selection	116	228	105
Extraction	3	27	75
Total	119	225	75
Percentage	26%	57%	17%

All the accepted papers were also evaluated to the classification criteria so it could be possible to make the study analysis, presented in the next section.

IV. DATA ANALYSIS AND CHARACTERIZATION

This section is dedicated to analyze data and answer the research questions based on the individual analysis of the mapping results and crossing data provided by classification criteria.

A. Use of Papers and Evaluation of Selection Quality

Figure 1 (a) presents results of the number of occurrence along the defined period of time, between before-selection and after-extraction phases. Observing the dark gray portion of the graphic it is possible to observe that the amount of papers selected after mapping kept fairly constant over the years. The observation of the difference between the light and dark area allows it to be noted that there was a follow-up between the evolutions of returned items and selected articles in the period analyzed and the difference was maintained between approximately 30 and 60 occurrences. Two observations can be made based on this information: (i) the Search String was compatible with the defined inclusion and exclusion criteria. That is, even before the application of the criteria, the oscillation of total papers returned along the years is almost similar to the oscillation of papers which relate to the research question in fact; (ii) the use of articles at the end of the mapping was 16% compared to the initial total. However, over the 449 initial papers, 116 were duplicated, that is, about 25%. Therefore, the actual use is approximately 22%.

B. Research Questions

The answers to the established research questions presented at Section II are discussed bellow.

R.Q.1. How publications about visualization tools and techniques, applied to software comprehension, have been evolved in the main digital libraries?

Figure 1 (b) presents a comparison of papers separated by digital library obtained from search string application before selection phase and after extraction phase. Analyzing the solid lines of the graphic, it can be observed that both before (light gray) and after (dark gray) criteria application, IEEE is the digital library with the largest amount of publications related to the search string and research questions along the years. It is interesting to notice the small difference between both solid lines, once it indicates the search string and criteria were well adjusted to this digital library. Another interesting feature to be observed is the oscillation along the years between IEEE and ACM, that is, periods when IEEE has a growth on the number of publications related to the research question, ACM had a decrease and vice-versa, and that this cycle recurred year after year.

The study identified 56% (42/75) papers as related to visualization tools which help at software comprehension. One was classified as experimental study. The others 32 relate to techniques, some with prototypes already implemented others still in conception stage. Crossing "Source" classification criterion (C.C.1) data with the "Focus" (C.C.2), it can be noted, at Figure 2 (a), that IEEE

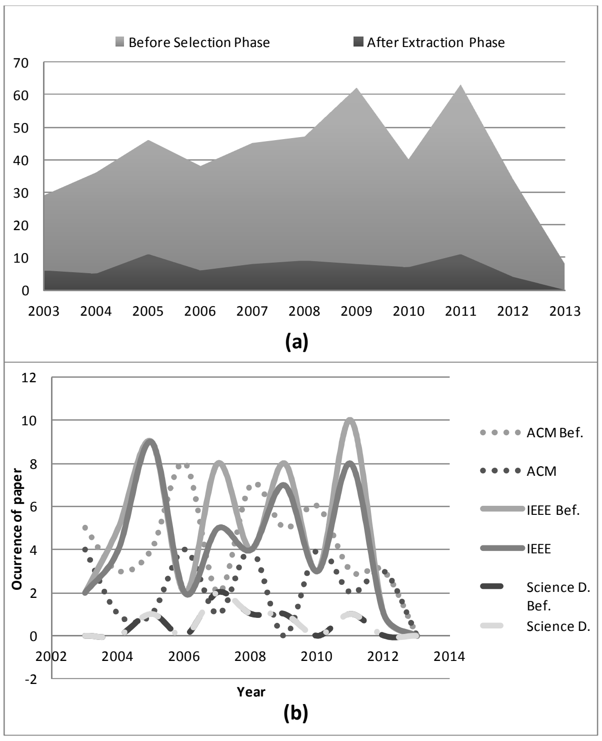


Figure 1. Graphic (a): Total ocurrence of paper before selection and after extraction phase; Graphic (b): The same information separated by digital library.

and ACM tool papers prevailed over those about technique, and Science Direct obtained the same quantity for both tool and technique. The only experimental study identified related to the research question [14] was published at IEEE.

R.Q.2. How visualization tools and techniques, applied to software comprehension, have been evaluated?

A total of 27% (20/75) of the tools/techniques does not demonstrate any kind of results of a practical evaluation, i.e., they do not indicate if the tool/technique proposed was tested to verify its efficacy/viability. From the others 73% (54/75), only 15% (11/75) were tested in a real development environment. Only 1.33% (1/75) paper was considered "Not Adequate". The other 58% (43/75) were tested in academic environment and had documented results.

Figure 2 (b) shows the values for "Evaluation Context" (C.C.3) along the years. Observing the prevailing area in light gray, it can be noted that evaluation context in academic environment (EAE) corresponds to the predominant form of evaluation, being followed by the non-evaluation of the proposal (NE). The evaluation in production context (EPC) remained modest, not exceeding 20% of the work until 2011.

This scenario reveals two important findings: (i) the studies need further validation to its practical viability; (ii) the test in academic environment has been the evaluation environment adopted in more than half of the studies but is not enough to represent the reality of a production environment which can mean a limitation on the external validity [15] of these proposals.

R.Q.3 What is the profile of visualization tools and techniques, applied to software comprehension, considering the artifacts represented, life-cycle phase and training?

More than half of the studies, 55% (41/75), considered the "Source Code" as the artifact used to generate the visualization. The Crossing between "Object of Analysis" (C.C.5) and "Analysis Criterion" (C.C.4) – Figure 2 (c) – allows identifying that "Static Analysis" corresponds to the kind of analysis most used, and the type of object that employs more this criterion in its representation is the "Source Code". The "Execution Tracing" corresponds to the object which most applies the "Dynamic Analysis" criterion. Also, few proposals of visualization apply both analyses.

A percentage of 81% (60/75) of tools/techniques defined corresponds to two-dimensional visualization, 18.7% (14/75) corresponds to tree-dimensional representations, 2.67% (2/75) employ both kinds of representation and one study was classified as "Not Adequate".

A total of 90.6% (68/75) of tools/techniques does not show any functionality to help the learning/training of new developers. Among other works, 8% (6/75) have focused on learning and one paper, 1.33%, is "Not Adequate" to any of the past options.

Observing the Graphic 6, generated by the crossing between "Representation Type" and "Learning Aided", it is possible to verify that the more used representation is the two-dimensional (2-D), which is about five times more employed than the tree-dimensional (3-D). Few visualization models adopt both types of representation, and regardless of the representation considered the learning aid remains little employed.

During the reading of the papers it could be observed that several tools/techniques identified do not allow the adequate scalability of the visualization. This problem attenuates when the tree-dimensional representation is used. The third dimension solves one of the problems of two-dimension visualization which is the lack of space. However, it seems to add a lot of complexity to the adopted visualization. Many were the times which was done the question "would this really help at software comprehension?" since exhibited examples are, generally, of hundreds of thousands of lines of code visualized in a small window. This may seems to be only a detail, considering the gains that these tools/techniques brings by its own existence, but the absence of this detail also is a limiting factor for an effective experience in user interaction.

These information can be used to delineate the profile of the papers: great part of these tools/techniques makes the static analysis of the source code, visualized in two-dimensional graphics with focus at help maintaining software. The reasons for the predominance of this pattern over others reveal an opportunity of research in this question, considering the causes involved. The few incidences of works with learning aid, reveals a deficiency and at the same time a potential research field. Aside it consists of a key factor for the acceptance of the new proposal by the starter user, the learning aid can accelerate and facilitate the training and conduction of experimental studies which evaluate, for example, the effectiveness of the same proposals, enabling their evolution.

V. THREATS OF VALIDITY

There are some threats of validity to our study. One of them is the fact the searches were made at only three digital libraries, which can restrict, in some aspects, the results. Some relevant digital libraries to the software engineering [16] were not consulted, due to time constraints imposed for the preliminary conduction of this mapping. In addition, the searches considered only results published in the last 10 years, and important papers may have been ignored even assuming these work evolutions were published inside the time window adopted. This approach was followed because the objective were to search tools/techniques which are being used currently, and it was considered that 10 years would be time enough to reach this objective.

It is also possible that we have chosen a search string which does not cover the whole set of relevant works. The software visualization field is somewhat wide and some important works, such as [17] [18], were not observed, due the fact of these papers do not use the key-words "comprehension" or "understanding" directly, for instance. The choose of classification criteria, although has been thought in a way to cover the high number of properties which characterizes this theme, can had ignored another characteristics as important and which collaborates to decrease threats related to the validity construction of the results [15]. It is also understandable that some papers, even adopting these terms, consider software visualization as a secondary focus on the software comprehension. Anyway, it was decided to follow with this string, once it returned the papers of control used, and so to guarantee that the results reflect our expectancy: papers about visualization focusing directly at software comprehension.

VI. CONCLUSIONS AND FUTURE WORK

As presented, this mapping work was divided in three parts: Planning, Conduction and Analysis. Conduction in turn, was subdivided in Study Identification, Selection and Extraction steps. The searches considered three important databases: ACM, IEEE and Science Direct and have identified initially 449 studies, from which only 75 were taken after applying the restrictions and selection criteria from Selection and Extraction phases. The data analysis lets the verifying of important information regarding how this research field has positioned with relation to the investigated

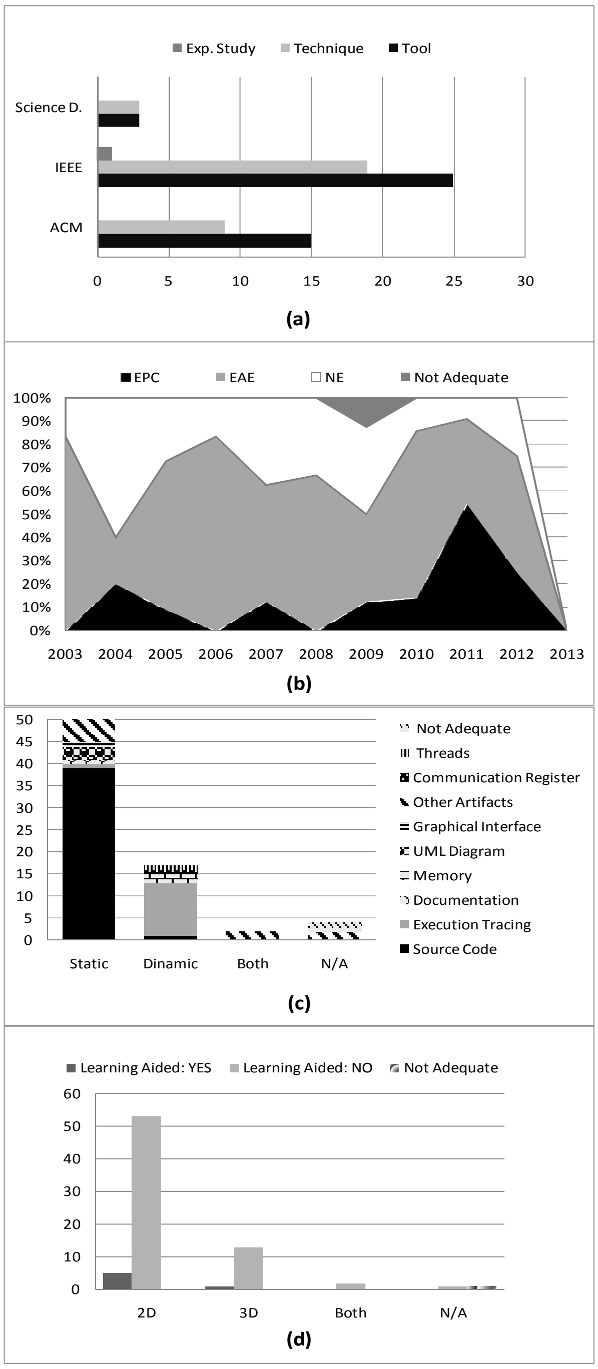


Figure 2. Criteria crossing for each graphic: (a) C.C.1 x C.C.2; (b) C.C.3 x Year; (c) C.C.4 x C.C.5; (d) C.C.6 x C.C.8.

theme. It was possible to identify, for example, a lack of tools/techniques of these works to support the training as well as the experimental validation of the pre proposes in the production environment. The preference for two-dimensional representation against three-dimensional and the lack of interest to questions related to interaction, considering the three-dimension, such as the question of the scalability, was another important point of this characterization. In addition, it was observed that the great majority of the proposes of visualizations adopts static analysis combined to source-code as software artifacts to be represented, which motivates the achievement of new experimental studies to investigate factors which influences these characteristics.

This work relates similarly to Bassil and Keller [13] in the sense that both are quantitative study, considering visualization tools related to software comprehension. Both works characterize some similar gaps and benefits in the field such as the importance of interaction issues and the visualization of source code aspects as being majority. The main differences are related to the considered source of information and the method applied. While the present work considers a systematic mapping, applied by software engineering researchers over scientific papers, the other was conducted as a survey with more than 100 participants (among researchers/users from industry) and considered a set of approximately 40 pre-defined tools.

This characterization study does not intend to be conclusive about the research questions investigated, considering the large extent of the covered theme and because we understand that some parameters of the protocol adopted such as the number of databases considered, researchers point of view, time windows established and other factors may let bias to the results. However, it can provide characteristic which still opened or are already consolidated and raises clues about cause-effect relations to the theme which deserves to be investigated.

In order to evolve the present work to future qualitative investigation, it should be considered visualization techniques to understand the importance and relationship of selected contributions to the research questions. Examples are the visualizations of co-authorship networks, citation graph, and impact ranking of author/paper.

Finally, this is a study based on an experimental process, yet established which lets the magnification/consolidation of the results through its replication. Artifacts generated in this study can be accessed in the following address [19].

ACKNOWLEDGMENT

The authors would like to thank the Instituto de Informática - INF/UFG, Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - CAPES - Brasil, and Fundação de Amparo à Pesquisa do Estado de Goiás - FAPEG - Brasil, which support this work.

REFERENCES

[1] S. K. Card, J. Mackinlay, and B. Shneiderman, Readings in Information Visualization: Using Vision to Think. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[2] R. Spence, Information Visualization: Design for Interaction, 2nd ed. London, England: Pearson Education Limited, 2007.

[3] P. R. G. Luzzardi, C. R. Andrade, and C. M. D. S. Freitas, " An Extended Set of Ergonomic Criteria for Information Visualization Techniques," in 7th Conference on Computer Graphics and Imaging, Kauai,Hawaii, 2004, pp. 236-241.

[4] C. Ware, Visual Thinking: For Design. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2008.

[5] C. Wohlin, et al., Experimentation in Software Engineering. Springer, 2012.

[6] J. Biolchini, P. G. Mian, A. C. C. Natali, and G. H. Travassos, "Systematic Review in Software Engineering," PESC-COPPE, UFRJ, Rio de Janeiro, Tech. Rep. ES-679/05, 2005.

[7] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in 12th International Conference on Evaluation and Assessment in Software Engineering, Bari, Italy, 2008, pp. 68-77.

[8] B. A. Kitchenham, "Guidelines for performing systematic literature reviews in software engineering," Keele University, Tech. Rep. EBSE-2007-01, 2007.

[9] E. Hernandes, A. Zamboni, and S. Fabbri, "Using GQM and TAM to evaluate StArt a tool that supports," CLEI Electronic Journal, vol. 15, no. 1, Apr. 2012, pp. 13-25.

[10] R. Wettel, M. Lanza, and R. Robbes, "Software systems as cities: a controlled experiment," in Proceedings of the 33rd International Conference on Software Engineering (ICSE '11), New York, NY, 2011, pp. 551-560.

[11] M. Lanza, S. Ducasse, H. Gall, and M. Pinzger, "CodeCrawler: an information visualization tool for program comprehension," in Proceedings of the 27th international conference on Software engineering, St. Louis, MO, USA, 2005, pp. 672-673.

[12] B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke, "A Systematic Survey of Program Comprehension through Dynamic Analysis," Software Engineering, IEEE Transactions on, vol. 35, no. 5, 2009, pp. 684-702.

[13] S. Bassil and R. K. Keller, "Software visualization tools: survey and analysis," in IWPC 2001. Proceedings. 9th International Workshop on, Toronto, Ont. Canada, 2001, pp. 7-17.

[14] A. R. Teyseyre and M. R. Campo, "An Overview of 3D Software Visualization," IEEE Transactions on Visualization and Computer Graphics, vol. 15, no. 1, Feb. 2009, pp. 87-105.

[15] T. D. Cook and D. T. Campbell, Quasi-experimentation: design & analysis issues for field settings, 3rd ed. Rand McNally College Pub. Co., 1979.

[16] P. Breretona, B. A. Kitchenham, D. Budgenb, M. Turnera, and M. Khalilc, "Lessons from applying the systematic literature review process within the software engineering domain," Journal of Systems and Software, vol. 80, no. 4, Apr. 2007, pp. 571-583.

[17] U. Erra, G. Scanniello, and N. Capece, "Visualizing the Evolution of Software Systems Using the Forest Metaphor.," in Information Visualisation (IV), 2012 16th International Conference on, Montpellier , 2012, pp. 87-92.

[18] M. Risi, G. Scanniello, and G. Tortora, "MetricAttitude: a visualization tool for the reverse engineering of object oriented software," in AVI '12, New York, NY, USA, 2012, pp. 449-456.

[19] INF/UFG. (2013, Oct.) Systematic Mapping Files. [Online]. http://www.inf.ufg.br/~auri/icsea-mv/

Managing IT Service Releases in a Systematic Way: A Case Study Approach

Marko Jäntti
School of Computing
University of Eastern Finland
P.O.B 1627, 70211 Kuopio, Finland
Email: marko.jantti@uef.fi

Antti Suhonen
School of Computing
University of Eastern Finland
P.O.B 1627, 70211 Kuopio, Finland
Email: antti.suhonen@uef.fi

Mika Kurenniemi
School of Computing
University of Eastern Finland
P.O.B 1627, 70211 Kuopio, Finland
Email: mika.kurenniemi@uef.fi

***Abstract*—Release management is responsible for planning, scheduling and controlling the deployment of releases to test and live environments. In many IT service provider organizations, the IT service release management is a very actual improvement target. Process frameworks, such as IT Infrastructure Library (ITIL), are often used as a basis of the process improvement. The research problem of this study is: How IT service releases can be managed in a systematic way? The main contribution of this paper is to present results of a case study with a Nordic IT service provider organization.**

Keywords*—*IT service; release management; process

I. Introduction

A *release* is a collection of hardware, software, documentation, processes or other components required to implement one or more approved changes to IT services [1]. Releases can be categorized into major releases, minor releases and patches. Release management activities should be conducted within the release management process that is coordinated by a release manager. A systematic approach for release management provides the following business benefits:

- delivering changes faster and at optimum cost and minimized risk [1]
- fewer releases to be rolled out to customers [2]
- releases are promoted successfully, are stable and meet expectations [3]
- releases are delivered according to agreed release policy and planned release cycles

There are three key challenges related to release management improvement from IT service management perspective. First, release management is often not seen as a process but is conducted in the form of separate activities, such as installations and packaging. This causes challenges for people who would like to improve the process because they cannot just go to employees and ask how they perform release management because employees do not know what is included in managing releases. Second, IT service organizations often lack the consistent understanding what is a release and how it is related to projects, service requests and change requests. Lack of understanding may lead to the following types of questions:

- Does a release cover installations required by a service request handling?
- Can we consider the project outcome of a deployment project as a release?
- Should every change implementation be treated as a release?

Third, a weak release management process typically leads to a fact that information on installations or releases is stored somewehere else than release records such as in change management.

Because ITIL is a best practice framework and not a standard, IT companies may aim at certifying their service management based on ISO/IEC 20000 standard family. The most popular parts of the standard family are ISO/IEC 20000-1:2010 Part 1: Service management system requirements [4] and ISO/IEC 20000-2:2011 Part 2: Guidance on the application of service management systems [5]. ISO/IEC TS 15504-8:2012 process assessment model [6] can be used to measure or improve the service management process capability. 15504-8:2012 provides the following base practices for release management [6]:

- Establish requirements for releases.
- Plan releases of services or service components.
- Design releases.
- Test releases.
- Deploy releases.
- Assure integrity of hardware, software, and other service components during deployment of the release.
- Reverse or remedy unsuccessful releases.
- Communicate release information to interested parties.

Much has been written about service management from service operation perspective. However, surprisingly few of studies have dealt with release management practices in IT service provider companies There are some studies that have focused on software release management such as the study of van Der Hoek and Wolf [7] that addresses requirements for release management: *...The release process should involve minimal effort on the part of the developer...The scope of a release should be controllable...*. Jansen and Bringkemper [8] discuss common misconceptions about product software release management.

Jokela and Jäntti [9] have identified challenges in release management process from product portfolio management perspective. They report that challenges were related to unclear release and deployment management and/or product portfolio management process roles, lack of process for product portfolio release and deployment management, lack of communication between product managers and lack of resources and time for product portfolio integration, testing and reviewing. There are studies that use the term patch management instead of release management, such as the study of Liu et al. [10] which presents methods for effective patch management. Jäntti and Sihvonen [11] have examined the patch management within release management. They observed that challenges exist especially in release management concepts and classifications. Patch management can be seen as a subprocess of release management.

A. *Our Contribution*

The main contribution of this study is

- to show how release management activities are performed in a Finnish IT service provider organization,
- to provide lessons learnt from release management process improvement.

The results of this study might be useful for release and deployment managers, installation team managers and other IT service management process managers. The remainder of the paper is organized as follows. In Section II, the research methods of this study are described. In Section III, case study results are presented. Section IV is the analysis of findings. The discussion and the conclusions are given in Section V.

II. Research Problem & Methodology

The case study was conducted during KISMET (Keys to IT Service Management and Effective Transition of Services) research project in May - June 2013. The research problem of this study is: How IT service releases can be managed in a systematic way? The research problem was divided into the following research questions:

- Which factors trigger the release management?
- How release management activities are performed in the case organization?
- What types of releases exist in the organization?
- How release management should be implemented with an IT Service Management tool?

A case study research can be defined as ”a research strategy focusing on understanding the dynamics present within single settings”[12]. Runeson and Höst [13] state that studies can be categorized into four types: 1) exploratory studies that focus on finding out what is happening, seeking new insights and generating ideas and hypotheses for new research, 2) descriptive studies that focus on portraying a situation or phenomenon, 3) explanatory studies focusing on seeking an explanation of a situation or a problem and 4) improving studies that aim to improve a certain aspect of the studied phenomenon. Our study could be classified as an exploratory

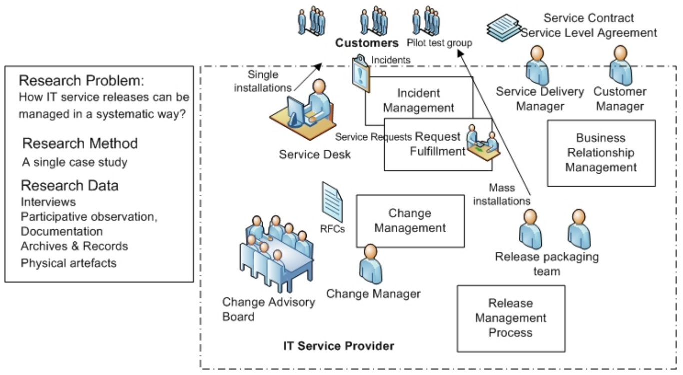


Fig. 1. The context of the case study

and improving case study. A case study research method with a single case was used to answer the research problem. Figure 1 shows the context of the case study.

A. *The Case Organization and Data Collection Methods*

Our case organization Alpha is a Nordic IT service provider company that provides IT outsourcing services and IT consulting services in Finland, Sweden, Norway and Denmark. Alpha has around 800 employees. The case study focused on exploring release management activities especially in workstation management service area. The company uses IT Infrastructure Library -based service management processes in incident management, problem management and change management. Release management was a natural choice for the improvement target because it is responsible for implementing changes.

The case study started with a kick-off meeting in May 2013 where improvement goals were discussed. The main objectives of the improvement pilot were to explore how release management activitities can be performed in practice, how release management could be implemented to the ITSM system and describe the process from a change to a release that is delivered to a customer.

Yin’s [14] data collection principles were used to increase the quality of the data collection: Data was collected by three researchers using multiple sources of evidence in Alpha’s facilities. A case study datastore was established and maintained during the study. Because NDAs were signed between a research team and the case organization, only three researchers were able to investigate the case study material. A chain of evidence was established by recording data sources (persons and their roles, date of data collection, document name) and linking findings to data sources. The following sources of evidence were used:

- Documentation (change plan, change task models, a list of standard changes, application package order form, image order form, workstation management service descriptions).
- Archives (Change request records, service request records)
- Interviews/discussions (change process owner, 2 change managers, CSI manager, release packaging team member)

- Participative observation (release management meetings)
- Direct observation (a Change Advisory Board meeting)
- Physical artefacts (Installation manager tool demonstration, access to development environment of the ITSM tool)

B. Data Analysis Method

The case study data was collected and analyzed by three researchers using a within case analysis technique [12]. Research findings were validated in two meetings with the representative of the case organization. The within-case analysis resulted in a case study writeup that was delivered to the case organization. The document summarized the case study findings and improvement actions.

III. IT SERVICE RELEASE MANAGEMENT: CASE STUDY FINDINGS

Next, a summary of the case study results is presented. In this paper, we focus on release management activities although the case provided a lot of findings related to the change management process.

A. Which factors trigger the release management?

We consider a Change Advisory Board and change managers as primary triggers for release management. The Change Advisory Board is a group of people that advises the change manager in the assessment, prioritisation and scheduling of changes [1]. Regarding authorization of changes we observed that change managers bring all the normal changes to CAB. In ITIL it is possible that a change manager may authorize the change without CAB meetings.

New standard changes are brought to CAB for preauthorization like in ITIL. After that they are typically handled in a service request fullfilment process. We observed that some installations are triggered by application package orders (Order form for application packages). A customer manager usually fills the form together with a customer and delivers the form to the service desk that submits the form to the packaging teams's queue. The order form for application packages defines the following details of the application to be packaged:

- Application name
- Number of users
- Application super user
- Application provider
- Description of application
- Storage for application media
- Installation code
- Language version
- Release method
- Operation system requirements
- Details of application package testing
- Target of release
- Change plan

Although these installations look like releases, it may be wise to exlude them from release management scope and record them as a part of request fullfilment process. However, normal changes that are processed by CAB could be scheduled and linked to a release. The case organization also seemed to lack the major change concept. We interpreted that a change with a major impact is equivalent to a major change.

There is a statement in ISO/IEC 20000-1 standard [4] that *requests for change classified as having the potential to have a major impact on the services or the customer shall be managed using the design and transition of new or changed services process.* A major change may occur in case of a new customer, a new customer for an existing service or a change that affects a certain number of users. Additionally, we may interpret that an emergency change is a change that receives the highest urgency level. We found an emergency change procedure in change management process description.

B. How release management activities are performed in the case organization?

The following observations were captured from the release management interviews with the case organization's employees:

- Change managers shall prepare the changes for the Change Advisory Board, a change manager can also reject a change.
- The biggest challenge is that there is no owner information regarding the computer the release should be delivered to.
- Customer might buy computers where we cannot put any images on.
- The request for a new release package may come from a customer through the service desk (application package form).
- If the form is poorly filled, a packaging team member shall retrieve the information.
- At the moment, Alpha does not have a change calendar.
- There are two tools used for installing software packages. The new one enables centralized installations, the old one requires establishment of customer site.
- Regarding the reports, customers are mainly interested in software usage level and application inventory (how many computers have a specific application version).
- Change and release schedules are agreed with customers by a customer manager / project manager / service delivery manager.
- Alpha has a small packaging team, thus a lot of issues shall be solved by discussions.

- An unsuccessful release is a release that fails to be installed to the computer. In case of a more complicated product, a user may inform the service desk that the application does not work.
- Major release updates shall be tested with all applications that need the update
- Alpha does not have a release note but information is stored in configuration management tool.

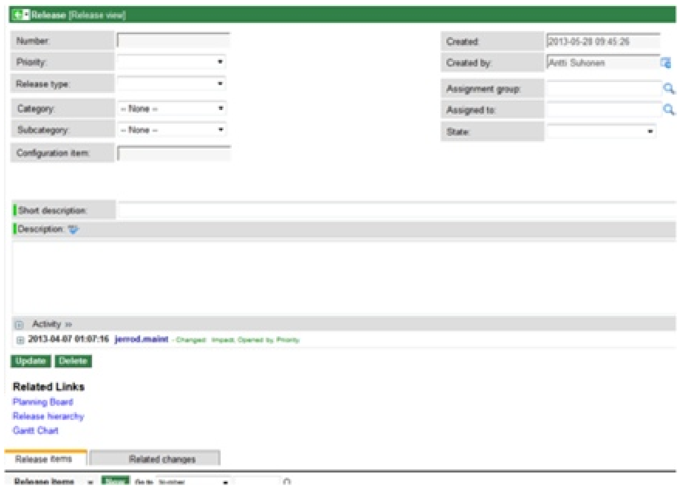
The following lifecycle for an installation was defined by the research team:

- Alpha's customer indicates the need for software distribution to a customer manager, or directly contacts the Service Desk (SD).
- Alpha's customer manager fills the billing information and submits the form to SD or SD fills out the order form based on the information given by a customer.
- SD controls the order of the packaging team queue in the ITSM tool (if the form has information gaps, the packaging group specialist calls for more information). packaging group builds a software package and tests it before using it.
- In order to deploy a release, the packaging group distributes the software package initially only specified customer (test) persons.
- If the distribution goes successfully to customers and they do not report any problems, then after a predefined time period distributions shall be done for all computers.

C. *What types of releases exist in the organization?*

Two different tools were used in the case organization to install software packages to customers. The research team participated in the demonstration of the new installation tool and identified the following types of releases:

- Audit (for example, google chrome updates)
- Configuration (java runtime environment, disable /enable java update)
- Critical updates (windows critical updates)
- Deploy (Windows program removal tool)
- Feature Pack (Windows patches, platform update)
- Hotfix (update for .Net framework)
- Microsoft unsupported (no more official support available for these releases)
- Rollup (collection of product updates)
- Security Advisory (single security updates)
- Security Hotfix (vulnerabilities in MS application)
- Security Update (application security updates)
- Service Pack (includes updates)


Release [Release view]

Number:		Created:	2013-05-28 09:45:26
Priority:		Created by:	Antti Suhonen
Release type:		Assignment group:	
Category:	-- None --	Assigned to:	
Subcategory:	-- None --	State:	
Configuration item:			

Short description:

Description:

Activity »

2013-04-07 01:07:16 jerrod.maint - Changed: Impact, Opened by, Priority

Related Links
Planning Board
Release hierarchy
Gantt Chart

Fig. 2. The draft version of the release record

D. *How release management should be implemented with an IT Service Management tool?*

The organization had recently changed their ITSM tool and had implemented incident management, service request management and change management to the new tool. However, the release management module had not been in use. One of the research team's tasks was to explore how release management could be implemented with a tool. Researchers spent a lot of time to look at change management module and its operational behavior.

Main observations from the tool side were the release module requires, for example, a button that enables creating a release from a change request, a user interface element that shows which change requests are related to a particular release, a release type field, release tasks that follow the release management process phases (for example, in planning, in testing) and finally hiding the Features. A Feature was a tool-related concept initially visible in release management user interface. A consultant from the tool provider side recommended hiding the concept to make the process simpler.

At the beginning, the difference between release items and release tasks was a little bit unclear to the research team. We interpreted that release items referred to the structure of releases and release tasks to the release management activities. Figure 2 shows the draft version of the release record.

At the end of the improvement pilot, the research team had a meeting with the ITSM tool development team. The result of the discussion was that most of the improvement ideas that the research team had suggested were implementable. The tool development team advised researchers to create RFCs to the Change Advisory Boad of the ITSM tool.

IV. Analysis

A within-case analysis technique was used in this study. This study showed that release management process improvement in IT service provider context is far away from a simple case. Release management process improvement is typically based on best practices of ITIL. It seems that the release management process is easier to be adopted by software providers than IT service providers. The following lessons learnt were derived from the case study.

Lesson 1: Strong change management affects the role of release management. When a change management process is deployed before release management, this may lead to a situation where change management may become relatively stronger process area than release management. There is a risk that keeping release management as a subpart of change management process, decreases the visibility of release management aspects. In our case, the most release-related information was stored in change tasks because there was no release record available.

Lesson 2: Transition of new or changed services is a complicated area. Transition of new or changed services is a process area in ISO/IEC 20000 standard [5]. This process area is related to release management in the following way: *...The transition of services should include the build, test and acceptance of the new or changed services followed by making the new or changed services operational through the release and deployment management process...*. We observed that both design and transition of new or changed services would have required clarification. We aim to clarify this issue by stating that building of a new service can seen as a major change. The implementation of a major change should be carried out as a project the outcomes of which form a release.

Lesson 3: Establish a release record. In early phase, we observed that there was no release record or release note practice in use. The release record could be visible to customers and show for example which incidents have been resolved by the particular release. In order to get change management and release management to support better ITSM best practices, a change record should have a field that allows the creation of a release. This Release button should be set visible not until the CAB has authorized the change. Basically, the button works in a same way than creating a problem record based on an incident.

The release type field may include four simple categories as a starting point: Major Release, Minor Release, Patch, and Fix. The release record should also guide the user to implement release according to predefined release tasks. The ITSM tool can be configured in such a way that a task needs to be completed before a new task can begin. To create traceability between installation tool and ITSM tool, one could add an action id of the installation to the release record of an ITSM tool.

Lesson 4: Implement a release schedule. One of our findings was that there was no clear release schedule that would show the frequency of releases. The research team recommended implementing a release schedule and communicating it to customers and staff such as service desk workers. There was evidence that some service areas in the case organization used maintenance windows that were communicated to customers.

Lesson 5: Define an emergency release procedure. The ISO/IEC 20000 standard requires that there is a documented procedure for managing emergency releases. We defined a very abstract level procedure:

- The need for emergency release is identified
- Every employee can make a decision on building an emergency releases
- Emergency releases shall be tested in a very light mode
- Emergency release shall be deployed to live environment
- Emergency release information shall be recorded in the ITSM system
- Emergency change shall be approved afterwards

Lesson 6: Assign a release manager role. According to our findings the organization does not have a release manager. Process managers have important roles both in ensuring that the process runs smoothly and monitoring and measuring the process. Sharifi et al. [15] have explored why ITIL implementations fail. One of the factors was not assigning process owners. The case organization should clarify who is responsible for the whole release management process. This role should be responsible for [2]: producing management reports, creating and maintaining release and deployment policies, providing reports on the progress of releases and ensuring that release management follows the organization's procedures and policies. A smaller organization might combine the role with a change or configuration manager role.

The above mentioned list is based on our findings from the case organization Alpha and lessons learnt are not presented in a priority order. This was the second case study on release management improvement with the case organization. In our first case study [9], the case organization had product-oriented business focus compared to Alpha that is a service provider. However, we observed same type of challenges, such as difficulties in defining a release policy. The main difference we observed was that in the product-oriented release management releases are defined by product features while in IT service release management releases are defined by requests of change.

V. CONCLUSION

The research problem of this study was: How IT service releases can be managed in a systematic way? The main contribution of this study was to explore release management activities in a Nordic IT service provider organization. The key improvement ideas we identified were related to classification of releases, understanding the difference between a release and a change request, release management coordination by a release manager, and implementing a release record to the ITSM tool.

This case study included certain limitations. First, data were collected by using qualitative case study research methods from one service area. Quantitative case study methods could have been applied to examine the number of failed changes and releases. Second, we used a convenience sampling as a case selection criteria. The research team had easier access to the case organization because they were an industrial partner of the research team. Further research could explore the release management interfaces with other service management processes such as configuration and change management.

ACKNOWLEDGMENT

This paper is based on research in KISMET project funded by the National Technology Agency TEKES (no. 70035/10), European Regional Development Fund (ERDF), and industrial partners. Special thanks to the case organization's staff for providing feedback, material and facilities to the research team.

REFERENCES

[1] Office of Government Commerce(c), *ITIL Service Transition*. The Stationary Office, UK, 2007.

[2] Office of Government Commerce, *ITIL Service Delivery*. The Stationary Office, UK, 2002.

[3] COBIT 5, *Control Objectives for Information and related Technology: COBIT 5: Enabling Processes*. ISACA, 2012.

[4] ISO/IEC 20000:1, *Part 1: Service management system requirements*. ISO/IEC JTC 1 Secretariat, 2010.

[5] ISO/IEC 20000:2, *Part 2: Guidance on the application of service management systems*. ISO/IEC JTC 1 Secretariat, 2011.

[6] ISO/IEC TS 15504-8:2012, *Information technology - Process assessment -Part 8: An exemplar process assessment model for IT service management*. ISO/IEC TC JTC1/SC7 Secretariat, 2012.

[7] A. van der Hoek and A. L. Wolf, "Software release management for component-based software," *Softw. Pract. Exper.*, vol. 33, no. 1, pp. 77–98, 2003.

[8] S. Jansen and S. Brinkkemper, "Ten misconceptions about product software release management explained using update cost/value functions," in *Proceedings of the International Workshop on Software Product Management*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 44–50.

[9] K. Jokela and M. Jäntti, "Challenges and problems in product portfolio release and deployment management," in *Proceedings of the 9th International Conference on Service Systems and Service Management (ICSSSM12)*. Shanghai, China: IEEE, 2012.

[10] S. Liu, R. Kuhn, and H. Rossman, "Surviving insecure it: Effective patch management," *IT Professional*, vol. 11, no. 2, pp. 49 –51, march-april 2009.

[11] H.-M. Sihvonen and M. Jantti, "Improving release and patch management processes: An empirical case study on process challenges," *Proceedings of the International Conference on Software Engineering Advances (ICSEA 2009)*, vol. 0, pp. 232–237, 2010.

[12] K. Eisenhardt, "Building theories from case study research," *Academy of Management Review*, vol. 14, pp. 532–550, 1989.

[13] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, no. 14, pp. 131–164, 2009.

[14] R. Yin, *Case Study Research: Design and Methods*. Beverly Hills, CA: Sage Publishing, 1994.

[15] M. Sharifi, M. Ayat, A. A. Rahman, and S. Sahibudin, "Lessons learned in ITIL implementation failure," in *Information Technology, 2008. ITSim 2008. International Symposium*, vol. 1, aug. 2008, pp. 1–4.

Pivots and Architectural Decisions: Two Sides of the Same Medal?

What Architecture Research and Lean Startup can learn from Each Other

Jan Salvador van der Ven
University of Groningen
Groningen, the Netherlands
mail@jansalvador.nl

Jan Bosch
Chalmers University of Technology
Gothenburg, Sweden
jan.bosch@chalmers.se

***Abstract* — Software architecture research has gained maturity over the last decades. It focuses on architectural knowledge, documentation, the role of the architect and rationale for the architecture decisions made. It is widely recognized that considering architecture decisions as first class entities helps in designing and maintaining architectures. In the entrepreneurial and new product development space, the lean startup movement is gaining momentum as one of the most notable ways to develop products. During new product development in highly uncertain environments, speed is the most important factor. Speed to get on the market, speed to learn from your customers, but also speed to tackle technological risks. Because the runway for new product development is short, it is important to experiment and make decisions quickly. The pivot plays a crucial role as a business decision for new product development. Both pivots and architectural design decisions can be seen as highly influential aspects for a product. In our research, we investigate what the fields of architecture research and lean startup could learn from each other. We focus our research on the two most important aspects of these movements: the architecture decision and the pivot, and show that they can be seen as two sides of the same medal representing the technical and the business side of the product.**

Keywords—Pivot; Architectural Decision; New Product Development; Lean Startup; Software Architecture.

I. Introduction

Every company changes direction multiple times during its lifetime. In the past, it took a company months or even years to change direction, especially in larger industry settings. In the last decade, the speed in which a company can adapt to changes has become one of the most competitive qualities [1]. The place where this effect is amplified is in new product development, either in small startups or in larger, established companies. Because these projects typically have a short runway to being successful, making decisions quickly is crucial.

Architects have the important role to align business strategy to the software architecture of the products [2]. Especially in the domain of new product development, this balance is an enormous challenge, because on the one hand the time to market is essential, and on the other hand the continuation of product and company is dependent on the solidity of the architecture. In new product development, there is also a bootstrapping problem. You need experiments with the Minimal Viable Product (MVP) in order to be able to validate your business assumptions, while you also need to have a piece of architecture to be able to create this MVP. This tension exists in many projects involving new product development.

As a software company, one of the most important aspects of your product is the software architecture, as it highly influences the capabilities (quality attributes) of the product. This architecture is formed by the decisions made during the development and maintenance [3]. Various authors emphasize the importance of these architectural decisions in software development [4, 5]. Models [6], classifications [7] and reasoning structures [8] have been posed to manage these decisions. Key concepts that are used in software architecture are: decision topic, rational, alternatives, choice, and risk.

Research literature studying new product development and startups [9, 10, 11] identifies a key type of decision that is extensively (and explicitly) used, the *pivot*. A pivot is the result of a business decision that is made to change the direction of the product. These decisions are based on different kinds of implicit or explicit experiments [1], in order to validate hypotheses about the product, its users or its business case. For the research described in this paper, we investigated what kind of decisions these pivots are, and what the relationships between pivots and the architectural decisions are. We currently focus on the pivots made at startups, because:

- At a startup, the runway is short, so the evolution of the architecture of the system is very high. Effects of pivots and architectural decisions are visible very quickly, and have a very high effect on the company's success.
- Larger companies are adopting startup techniques [1] to increase their own time-to-market, especially for new product development. This makes our research relevant as learning for large companies seeking new product development.

The contribution of this paper is threefold. First, we introduce a conceptual framework for new product development as an experiment system with pivots and architecture decisions as first class entities. Second, we identify the key concepts for architecture research and new product development, and identify the gaps between them. Third, we provide guidelines for the two fields that describe what they could learn from each other, based on the conceptual model and the identified concepts from both fields.

This paper is organized as follows. First, we introduce our conceptual framework. Then, we sequentially describe the concepts of software architecture (Section III) and new product development (Section IV) from a research and a practical perspective. In these sections, the key concepts are identified. Then, we describe the differences and similarities of the two as analysis in Section V. Based on this we present our guidelines for both fields. This paper ends with related and future work and some concluding words.

II. Conceptual Model

The premise of this paper is that this experimentation, both in the business domain and the technical domain, is a critical technique to increase the chances of success in new product development. Based on our literature findings, we have constructed a conceptual framework for running new product business as a set of decisions. In Fig. 1, our conceptual framework is visualized. On the top, two essential risks are shown as input for the business: market risk and technology risk. Which risks are most important depends on the context: the problem addressed, the market, the competitors, the solution chosen, the technical possibilities, etc. Based on which risks are most eminent, hypotheses are formulated to reduce uncertainty of the associated risk. To test each hypothesis, one or more experiments are performed. These experiments can be explicit (e.g., conducting a planned usage test, running a Proof of Concept, predict usage statistics), or implicit (e.g., a coincidental encounter, different product use by end-users). Then, based on the results of the experiments, decisions are made for the direction of the product. These decisions steer the direction of the product and the associated business, affecting the market and/or the software architecture, and in the end the product itself. In new product development, pivots are illustrative examples of these decisions. Therefore, the naming of the decision types is based on the phases described by Maurya [11]. In the initial Problem / Solution (P/S) fit stage, the decisions don’t affect the system at all, since there is typically no product yet. In the second, Product / Market (P/M) fit stage, the focus of the experiments is to validate the Minimal Viable Product. This can result in pivots that influence the business as well as the product. For these decisions, the market fit is the most important; so, the architectural impact is subordinate. In the following phase, assuming that the product / market fit is validated, still experiments need to be conducted to figure out how to scale the product when usage (e.g., number of users or usage per user) grows. Aside from direct business requirements, in each stage software architecture decisions need to be made, for example to support increasing scale, reduce technical debt or support an alternative use case after a solution pivot. This paper focuses on pivots as decisions that arise from experiments that affect the business as well as the architecture of a product.

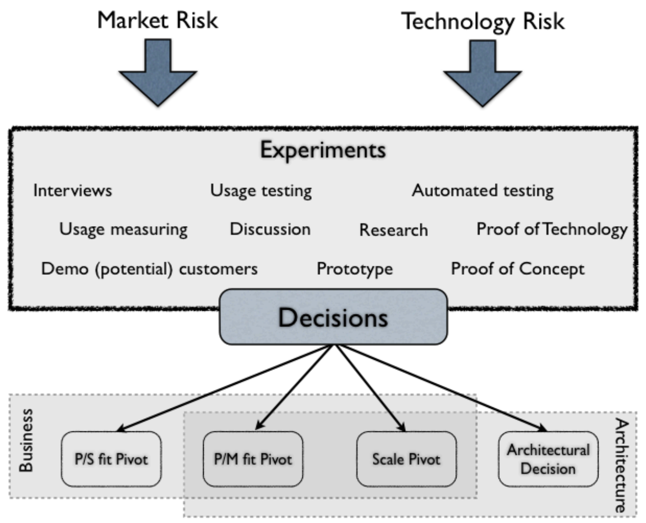


Fig. 1. Graphical representation of the conceptual framework for decision-based new product development.

The validation speed is very important in this context. Validating a hypothesis takes time and effort. This effort should result in new insights in the product or the market. If the product changes direction later (a pivot, or abandoning a pivot), the effort should pay itself by what is learned by it. So, it is important to keep validation speed short, and create hypothesis focused on learning. This is why validation speed it essential in our model.

When looking at product development through our conceptual model, it is possible to see that pivots and architectural decisions are actually the ways to mitigate risks by experimentation. However, they both have a different risk they are addressing, while affecting each other constantly. So, they can be seen as two sides of the same medal, one side showing the market challenges, while the other side shows the associated technological risks. It is virtually impossible to encounter one without the other, as market risks are typically tackled with technological solutions (e.g., the business drivers for the architecture), and technology always affects the business.

In the following sections, we will describe how this model can be used in both software architecture and new product development.

III. Software Architecture

A. *Software Architecture Research*

Software architecture has been researched extensively in the last decades [12, 13, 14]. In this research, architectural knowledge [6, 15] and more specifically architectural decisions [5, 16, 17] play a vital role. What we can distill from this research is that creating architectures is essentially a risk-mitigation process where the balance has to be found between non-functional requirements (e.g., quality attributes), business risks and technological challenges. Often the long-term view is more important then short-term project goals for making the right architectural decisions. In high-pressure situations (e.g., deadlines), it is easy to give in on these long-term issues, causing design erosion [18], technical debt [19] or even worse, project failure. In the next section, three cases are described that show how architecture decisions are used in practice. From this, we identify key concepts for comparing architectural decisions and pivots.

B. *Cases*

In order to be able to compare software architecture practices to lean startup movement, we have to identify what parts are eminent for both fields. To do this for the software architecture space, we have conducted a literature research combined with our experience as participant researchers in several cases. We have analyzed the practices of software architecture in new product development in several cases [7]. In this paper, we summarize the cases that contain relevant information about how software architecture is used in practice. The cases are anonymized to protect the companies and customers involved. The cases are not selected at random. From the experience of the authors, other cases could have been chosen. However, as Eisenhardt [20] poses, in case study research it is ”neither necessary nor preferable to randomly select cases”. We have chosen to discuss the cases that considered new product development, while being large enough to be relevant as industrial cases. A more extensive description of these cases can be found in our previous work [7], where we focused on the role of the architect in the software development process. In this work, we describe our findings of these cases that consider pivots and architectural decisions.

Case Alpha involved the construction of a software system that had to replace a legacy Geographic Information System (GIS) for a large harbor. The new system had to be coupled with several legacy backoffice systems. The customer, a large harbor company in the Netherlands, initiated the project. The solution was service oriented, and consisted of several systems communicating with each other through an Enterprise Service Bus. Most of the software was written in Java. The coupling was one of the most challenging issues in the project. This case consisted of a pilot and a realization phase, three and six months, respectively. Ten to twenty people were involved during the various phases of the project.

Case Alpha was a typical example of a project that was driven by risk management in order to get the architecture of the system right. Several techniques were used to experiment in order to mitigate risks. In the pilot phase, the time was fixed, and the goal was to show the most important (technical) risks could be tackled. This resulted in a biweekly iteration that focused on tackling the top-priority risk. In this phase, a PoT (Proof of Technology) and a PoC (Proof of Concept) were made, involving many architectural decisions. Both the PoT and the PoC were demonstrated to the customer as well as the end-users to validate critical assumptions.

Case Beta was conducted at a medium sized product company in the Netherlands. The project involved a new administrative software system for specific departments in Dutch hospitals. Changing regulations and different working environments needed to be taken into account. The project was executed by a multidisciplinary team of seven people, assisted by the architect from the company. A Java stack (JSF, Spring, Eclipselink) was used for creating this product from scratch, while a different team of approximately seven people developed a part of the backend separately. This separate development was one of the most challenging architectural parts of the project. The development of the product took place for a period of 12 months.

In case Beta, several architectural experiments were conducted, the major one consisting of how to manage the introduced complexity of the platform. A prototype was constructed early on. Also, interviews were held with key users in the field. However, often the experiments were conducted ad-hoc without a concrete hypothesis to validate. The architectural question if the generic backend part of the system could be reused was validated continuously by using this component in another project, too.

A small startup company working on a web based product for the consumer market was the scene for case Gamma. The project contained high-risk technological challenges, where the architecture needed to be flexible in the beginning, to be able to handle the expected high number of users. The application was created in Ruby on Rails[1] with a NoSQL backend based on MongoDB[2] and Redis[3]. The main architectural challenges were to be able to potentially scale up the application when lots of consumers are using the system, while being able to adopt the system to changing requirements from the customers.

Case Gamma consisted of constant experimentation. As the product of the company was being developed, several hypotheses were considered, resulting in either small pivots (e.g., users would like to see the results in a stream-like view), or architectural decisions (e.g., the graph database could be best modeled in Redis). However, again the experiments were setup implicitly, e.g., without forming a hypothesis or validating if the results were expected.

We have seen the experimental nature in all of these cases. Also, in all of the cases a clear Build, Measure, Learn (BML) loop [10] was used. In cases Alpha and Beta, this loop was used implicitly (never mentioned), while in case Gamma the BML loop was known and explicitly used.

C. Key concepts

Several key concepts come back in most of the research about architectural decisions [21]:

- **Architecture Design decision.** Design decisions are the building blocks for software architecture. These decisions consist of the following parts:
- **Decision topic.** The decision topic is the actual problem that needs to be solved. Often, these topics arise from previous decisions (we decided to base our application on NoSql technology, which specific database product are we going to use?), or from non-functional requirements (how are we going to ensure our up- time is high enough?)
- **Choice.** The choice, or decision, is the result of the decision process. Often, this is the only part that is communicated (discussed or documented).
- **Alternatives.** A typical decision has more than one alternative to chose from. Alternatives can be just named (e.g., different component names), or sometimes architecture parts are considered as alternatives (different styles or patterns, or comparing specific implementations of components). In rare cases, the alternatives are realized and compared as a Proof of Concept or Proof of Technology.
- **Rationale.** The rationale of a decision describes, often in plain text, why the chosen alternative(s) solve(s) the problem at hand, and why the chosen decision is the best solution.

Based on our case material, we have seen two other key concepts that are important around software architecture design decisions:

- **Risk.** Decisions are often made to mitigate a risk. So, in order to address a concrete market or technological risk, certain decisions need to be made. Risks can be seen as triggers for decision topics.

[1] http://rubyonrails.org/

[2] http://www.mongodb.org/

[3] http://redis.io/

- **Experimentation.** To make sure you make the right decisions often, besides the rationale already discussed, experiments are conducted to make viable that the suggested solution is correct. This can be done either as a PoT, PoC or something else.

In the following section, we will describe what the nature of new product development is and how the lean startup movement influences it.

IV. New Product Development

A. Research

Experimentation in Research and Development (R&D) as a basis for decision-making is the normal approach in a variety of domains, including the manufacturing, automotive, mechanical engineering, medical, and pharmaceutical industry [22]. From the experiential perspective, frequent iterations of products in terms of prototypes or multiple design iterations, testing, and more frequent milestones are associated with faster product development [23]. In the software industry, innovation through experiments with customers is becoming more and more discussed [24], primarily in the web 2.0 and Software as a Service (SaaS) fields. However, in the software industry, these experiments are currently primarily performed in pilot stages for validating architectural decisions or on feature optimization.

B. Interview Setup

We have conducted interviews with founders and architects of startup companies, to identify what pivots were made in new product development, and what the nature was of these decisions. In our interviews, we have chosen to focus on pivots as an entrance to talk about the most important decisions and the decision process. We interviewed representatives of the five different companies. In these interviews, we discussed a total of nine pivots. Two of the companies were located in the Netherlands, two in the USA, and one in Sweden. All the companies were product companies, delivering web-based software.

As our research has an exploratory nature, we have chosen to use semi-structured interviews for acquiring our data. The interviews lasted from one to two hours. We have recorded all of the interviews to be able to listen again to the conversations during the analyses phase. In addition to this, the interviewer made notes during the interview. Based on the notes and the recordings a log is created with results after the interview. These logs were the basis for our analyses.

The interviews were structured as follows. First an introduction was given about the current status and the goal of the research. The interviewee was asked permission to publish about the results and if it was okay that the interview was recorded. Then general questions about the company and terminology was asked, after which the interviewee was asked to tell about several pivots he was involved in. The interviews in the Netherlands were done face-to-face in Dutch, while the interviews with Sweden and the USA were done via videoconference in English.

We have used interview questions as guidance through our open-ended interviews. First, basic questions about the interviewee and company were asked, including if the company worked according to lean startups principles and if the architecture of the system was considered explicitly. In order to relate the results of the different interviewees to each other, we have asked them to describe what they mean by three key terms in our research: *pivot*, *architecture*, and *architecture decision*. Then, we used a set of questions to let the interviewees reason about the their pivot. As we wanted to focus on the decision process around pivots, we have not extensively questioned the technical details, but focused on the decision part of the pivots. The interview questions that were used are shown in Table I.

TABLE I. INTERVIEW QUESTIONS

Question
Can you give a short description of the pivot?
Who were involved in the decision process?
What triggered the pivot?
Did you validate the success / results of the pivot? How did you do that? How long did it take to do this validation?
Were there any alternatives evaluated? If so, what alternatives?
What were the results of the pivot?
Did the pivot affect the (software) architecture of your system / product?
What were the results on the architecture?

These questions were used as a baseline for the interview. Where viable, additional questions were asked, or explanation was asked for. In some cases, when the answer to a question was already told or when the question was irrelevant for the context, the question was skipped and later noted based on the recordings and notes.

For our research to be generic, we have selected a variety of interviewees and companies. On the other hand, we had to narrow our research in order to make sure the interview results would be comparable. We used the following criteria for selecting the companies:

- Companies from software industry in the startup phase, or a close startup origin.
- Companies at least one year in business at the time of the discussed pivot(s).
- Companies that produce a product or service (no consulting).
- Companies with more than one employee.

TABLE II. OVERVIEW OF COMPANIES

Company	Location	Role	Domain	Size
Voys	NLD	Founder	Voice over IP, telecom for small business	~23
Certive	USA	Lead Engineer	Enterprise analytics software	~20
Data-provider	NLD	Founder	Data	~10
Burt	SWE	Chief Architect	Analytics for publishers	~28
Zevents	USA	Lead Engineer	Local search advertising	~50

This resulted in the selection of a set of 5 companies, as shown in the Table II. In the columns, the Company name, the

TABLE III. OVERVIEW OF PIVOTS

#	Comp.	Pivot / Decision	Prioritized risk (type)	Experiments and validation	Alternatives evaluated	Results on architecture
P1	Voys	Business model change	Unknown	Accidently showing internally used functionality to a customer.	None	The architecture became more of a 'Christmas tree'
P2	Voys	Architecture reconstruction	Maintainability decrease	Technological exploration	1) Buy functionality from other suppliers and 2) merging with other company	Reworked architecture, the system was now manageably growing
P3	Voys	Change of product packaging	Customers misused the product	Usage testing and measuring	None	Unknown (currently in progress)
P4	Certive	Radical change in business	Unknown	Demonstrating a mock-up to potential customers at a conference	Unknown	Moved more to hosted and cloud-based services
P5	Data-provider	Scaling the indexing possibilities	Technical possibility to scale product	Technological pilots, automated performance validation	All different kinds of NoSql solutions were evaluated	Possible to index sites at a high speed.
P6	Data-provider	Enhance defect efficiency	Data not accurate enough	Usage Measuring and experimentation at customer site	1) External provider for data and 2) buying data from others	Not much, the major change is in the way the application was used (the customer can decide the error rate)
P7	Burt	Change of customers from advertisers to publishers	Advertiser market is uncertain business	Usage measuring and discussion	1) Stay on advertisers and 2) move to both publishers and advertisers	Better distributed scalable architecture. Many principles were decided on (e.g., start with two on anything)
P8	Burt	Change in product from advertiser tool to analysis tool for advertisers	Customers are not able to judge the market value of product	Prototype, Demonstrate to potential customers	Several prototypes of different ideas were tried	Change from desktop to web based platform
P9	Zevents	Change in focus on search instead of publisher oriented site	Business of publisher sites was going down.	Discussion, prototypes	Lot of discussion about other alternatives tool place. One alternative was offering 'deals' to for local companies.	Architecture and tooling became more 'generic', making it harder for the company to distinguish itself against others.

geographical location, the role of the interviewee, the domain of the company and the size of the company (number of employees) is described. From each of the companies, we interviewed one of the key persons involved in the pivot(s) that occurred.

C. Interview results

First, we had to identify our interviewees' point of reference. To do this, we asked them about what three key terms in this research mean to them.

- **Pivot.** Even thought the term pivot is widely used in software industry, there was some difference in the explanations about what a pivot is. Two points came back in all interviews: that it is a radical interruption against the 'previous' way of working/thinking and that often, different users/customers were targeted after a pivot. So, the business strategy of a company changed. One person emphasized that layoffs are often the result of a pivot, making it 'scary' for employees when a pivot occurs.
- **Software Architecture.** The traditional view on architecture was dominant at the interviewees. All of them identified connectors/interfaces as one of the most important parts of architecture. Also, the mapping of business (requirements) on the technical design of the system was mentioned often.
- **Architectural Design Decision:** One of the interviewees had no idea what an architectural decision meant. The others noted that it is a conscious decision, where a specific direction is chosen for the architecture of a system (a branch-point).

We have summarized the results from the interview in Table III. In this table, after the name of the company and a short description of the pivot, the risk that was tackled by the pivot is described. The next column describes what experiments were conducted to validate the pivot. This information was derived from what the interviewees discussed based on the interview questions (e.g., the trigger for the pivot and the alternatives evaluated). Then, evaluated alternatives are shown, and in the last column of the table the results on the architecture are described.

Although Ries [10] identifies ten different types of pivots, he does not discuss the effects that pivots have on the architecture. From our interviews we have found that it is possible to typify pivots by the impact they have on the architecture, as described in our conceptual framework. Business (product/market fit) pivots were found in six of the pivots and scale pivots were identified in three of the pivots. Although all interviewees stressed the fast-paced, dynamic and uncertain nature of new product development, the importance of employing a structured, systematic approach to decision making was recognized as important.

D. Key Concepts

The following key Concepts involving new product development are extended from the literature:

- **BML / Experiment.** The basis of the lean startup lies in the Build Measure Learn (BML) loop, as described by Ries in [10]. This means that in order to find a sustainable business, one has to continuously execute experiments (build), measure the effects, and learn from the results.
- **MVP.** The Minimal Viable Product (MVP) is the first version of the product that can be used to start the BML loop. This can be a first version of a product, but it can also be something else (e.g., a landing page, video) as long as the hypotheses about the product can be validated.
- **Hypotheses.** In order to be able to know if one goes in the right direction, you have to know where you want to go. This is posed in a hypothesis that can be tested by experimentation.
- **Validation.** Key to understanding the results of a build step is to identify how to validate or invalidate a hypothesis.
- **Measuring.** Even though validation is concerned one of the most important parts of the BML loop, the measuring is always an arduous part. Measuring can be done either qualitative (e.g., interviews), or quantitative (surveys, usage measuring, A/B testing).
- **Pivot.** A pivot is a key concept in the lean startup movement as a decision to change direction for a product. Several types of pivots have been identified by Ries [10].

Based on the interviews, an additional concept comes back:

- **Risk.** Most of the pivots that were discussed in the interviews mentioned that they were done in order to mitigate some risk. The identification of this risk was often the starting point for the pivot.

V. Analysis

In this section, we summarize what similarities and differences are between the architecture research space and the startup spaces, by comparing the most characteristics aspects of both: architectural decisions and pivots. The introduced concepts of both software architecture and lean startup / new product development are compared in Table IV.

One of the biggest differences is the focus. As the architecture community focuses on long-term non-functional requirements, the lean startup community focuses on rapid validation of business assumptions (hypotheses). This also has a cost implication. For lean startups, the speed of validation is the most important aspect. So, the experiments should be as fast and cost-efficient as possible, to be able to change direction quickly if market or technology demands that. This contrasts the approach of the architecture community where the focus is much more on making correct decisions to reduce cost later in the development.

Several parts come back in both worlds. Both consider risks as primary triggers for making a decision, and both have an explicit description of what needs to be solved, the decision topic and the hypothesis. Further, both parts use experimentation to see if the decision is correct, even though these experiments have different forms. The minimal version to validate your decision is correct also comes in different forms, in architecture this is often a technological proof while in new product development this typically involves customers and end-users.

Further, as can be seen from the table, several concepts from one field seem to be nonexistent in the other field. The

TABLE IV. COMPARING CONCEPTS

Architecture Decision Concept	Lean Startup Concept	Software Architecture	New Product Development
Architectural Design Decision	-	First class entity for the architecture	-
-	Pivot	-	Radical change in business model
Decision topic	Hypotheses	Decision topics are typically hierarchical (caused by previous decisions), or caused by arising or expected risks.	
Choice	-	Often referred to as the decision self, this is the selection of the best alternative	The choice is not explicitly mentioned in new product development space.
Alternatives	-	Are often made explicit in documentation	Alternatives are rarely made explicit.
Rationale	-	Existing in the heads of the developers, or (ideally) written down explicitly	Less relevant as the results are measured quickly.
Risk	Risk	Often the focus is on technological risks. Is addressed by reasoning, often the cause of an decision topic and thus a design decision	Focus is on the business risks. Is addressed by experimentation
Experimenation	BML / Experimentation	Automated testing (e.g., performance tests), Research, Discussion	Interviews, Usage measuring, Demonstration, Discussion, Prototyping, Research, Usage testing
PoC / PoT	Minimal Viable Product	In order do address certain risks, PoCs or PoTs are conducted. Main goal is to validate the viability of the concept or the technology, not the business	One of the main goals for a product under development. Main goal is to start validating the business model as quick as possible.
-	Measuring	Rarely done	Measuring is the only way to validate the hypotheses
-	Validation	Is often not done, if it was done, it was done by reasoning. It is often hard to validate a NFR	Direct business validation. Often the existence of company validates pivot.

explicit parts of the decision in the software architecture field (Choice, Alternatives, Rationale) do not exist in the Lean startup field. Alternatives are evaluated (as seen in the interviews) and rationale is used to argument decisions or pivots, but decisions as first class entities are not common in the lean startup field. On the other side, the measuring and validation that is key in the lean startup is not considered in the architecture space.

A. Threats to Validity

This research is based on a limited set of cases and interviews. To a certain extent interviews bare some subjectivity in them, because it is a conversation between two individuals. Because of the exploratory nature of our research, using semi-structured interviews was a good way to validate our model. However, this research could be extended by more interviews, and by gathering more quantitative data based on surveys, as described in the future work.

For interview validity reasons, we have not presented our framework or model to our interviewees. This would have biased our interviewees, and perhaps changed the way they described the pivots, and answered the questions.

VI. GUIDELINES

In addition to confirming the conceptual framework, the data presented in this paper allowed us to derive a set of guidelines about what the field of software architecture and new product development could learn from each other.

A. Solve both business and architecture as experiments

For new product development, explicit experimentation is common. Architects can learn from this by doing similar explicit experiments to validate the architectural decisions at hand. This helps architects to speed up development and develop business quicker.

B. Business as a set of decisions

As shown in our conceptual model new product development can be treated as an iterative process of running market and technology experiments. The experiments are driven by the risks that need to be tackled, and the result of the experiments is a set of decisions that form the business and the product. As we have shown that an architecture can be seen as a set of decisions, we think this view can be extended when considering pivots as business decisions. In this view, the business can actually be seen as the set of taken decisions based on the results of experiments.

By making the decisions in new product development more explicit, it is possible to piggyback on the experience that the software architecture research already developed. It can for example be used to trace the decision process, change decisions when the situation changes, and see the dependencies that decisions have on each other.

C. Creative validation of architectural decisions

Even though some efforts are made to validate architectural decisions, the field of software architecture could benefit much from the creative way that lean startups validate their hypotheses. Of course, the horizon for both decisions is not always the same, but the tendency to validate an architectural decision by reasoning could be enhanced by more objective ways of validation (e.g., usage statistics, A/B testing).

D. Sometimes, architecture can be added later

We have seen that in highly uncertain environments pivots affect the balance in the development of new products. Since p/m pivots put the emphasis on validating the business, the architecture of the product is often minimal supported. This can cause design erosion and technical debt. However, we have seen that there are several strategies used at our investigated companies to overcome this:

- **Pivot away.** The first strategy we identified was that in some cases the pivot was so radical, that the current architecture was thrown away. So, no matter how unbalanced the scale was, the complete business changed and the complete architecture of the system changed too. Off course the experience of the team and the business knowledge is reused, but the system itself was largely or completely rebuild. Sometimes a complete new technology stack was adopted (P2, P4, P8), while in other cases existing components were reused (P1, P5, P7).
- **Add architecture later.** When a product/market fit is found, but the architecture of the system is unable to facilitate the next phase (scale, as described in [11]), then architecture needs to be added later. So, in order to handle certain (non-functional) requirements for scaling, like performance or changeability, the architecture of the system need to be improved. As we have seen in our interviews (P2, P5), this is possible even though it can be expensive.

VII. RELATED WORK

Although the field of new product development is not new, lean startup is quite new, and within the research community there has not been much research about this topic. The basis for our model, experimentation, lies in the work of Thomke [25] and Davenport [24]. This was extended with the methodologies from the lean startup community [9, 10, 11]. From our own work on architectural design decisions, we generalized the idea of running a business as an explicit set of decisions [7], based on the experiments [1].

The relationship between business and architecture has been extensively studied from the product line perspective, for example BAPO [2]. We have shown that two types of decisions are extremely important in new product development: business (e.g., pivot) and architecture decisions.

VIII. FUTURE WORK

Based on the encouraging results from our research, we are planning to extend it in several ways. First, we are planning to interview more people, to extend our data set and further validate and refine our findings. For example, we have not had any of the interviewees talk about hypotheses, even though the literature emphasizes hypothesis-based experimentation. Second, we are planning to extend our question set to a questionnaire that can be send to a larger group of people for a more quantitative validation.

Also, we are planning to test the usage of our model in industrial settings. For this, we are planning to conduct case studies at several companies, where we would guide the company into using the conceptual model, and reflect on the efficiency. This could sharpen our framework and it would give further validation of the viability of our proposed work.

Last, we would like to extend our guidelines to even more actionable guidelines that could be used in the various stages a product can be in.

IX. CONCLUSIONS

In this research, we have shown that new product development is based on two types of decisions: architectural decisions and pivots. We have presented a conceptual framework that addresses both decisions in the context of an experimental risk-based process. This framework can help practitioners to structure their new product development process. From our interviews we derived a set of guidelines that emphasized the importance of decisions in experiments. Both architectural decisions as well as pivots play a vital role in the development of new products, as two sides of a medal representing the technical and the business part of a decision.

ACKNOWLEDGEMENTS

We would like to thank the following interviewees for taking the time to talk with us about their pivots: Mark Vletter, Gordon Rios, Christian Branbergen and Theo Hultberg.

REFERENCES

[1] J. Bosch, “Building Products as Innovation Experiment Systems”, in ICSOB, Springer, 2012, pp. 27-39.

[2] F. van der Linden, J. Bosch, E. Kamsties, K. Känsälä, and J. H. Obbink, “Software Product Family Evaluation”, in SPLC, Springer Verlag, 2004, pp. 110-129.

[3] A. G. J. Jansen and J. Bosch, “Software Architecture as a Set of Architectural Design Decisions”, in Proceedings of the 5th IEEE/IFIP Working Conference on Software Architecture (WICSA 2005), IEEE Computer Society, 2005, pp. 109-119.

[4] J. S. van der Ven, A. G. J. Jansen, P. Avgeriou, and D. K. Hammer, “Using Architectural Decisions”, in Second International Conference on the Quality of Software Architecture (Qosa 2006), Karlsruhe University Press, 2006, pp. 1-10.

[5] P. Kruchten, “An Ontology of Architectural Design Decisions in Software Intensive Systems”, in 2nd Groningen Workshop Software Variability, 2004, pp. 54-61.

[6] R. C. de Boer, R. Farenhorst, P. Lago, H. van Vliet, V. Clerc, and A. Jansen, “Architectural knowledge: getting to the core”, in Proceedings of the Quality of software architectures 3rd international conference, 2007, Springer-Verlag, 2007, pp. 197-214.

[7] J. S. van der Ven and J. Bosch, “Architecture Decisions: Who, How and When?”, in ASA, Agile Software Architectures (to be published), Elsevier, 2013, pp. unknown.

[8] D. Tofan, M. Galster, and P. Avgeriou, “Reducing Architectural Knowledge Vaporization by Applying the Repertory Grid Technique”, in Proceedings of the 5th European Conference on Software Architecture (ECSA), Springer LNCS, 2011, pp. 244-251.

[9] S. Blank, “The Four Steps to the Epiphany: Successful Strategies for Products that Win”, Lulu.com, 2008.

[10] E. Ries, “The Lean Startup: How Constant Innovation Creates Radically Successful Businesses”, Penguin Books Limited, 2011.

[11] A. Maurya, “Running Lean: Iterate from Plan A to a Plan That Works”, O'Reilly Media, Incorporated, 2012.

[12] J. Bosch, “Design and use of software architectures: adopting and evolving a product-line approach”, ACM Press/Addison-Wesley Publishing Co., 2000.

[13] C. Hofmeister, R. Nord, and D. Soni, “Applied Software Architecture”, Addison-Wesley, 2000.

[14] P. Clements, D. Garlan, L. Bass, J. Stafford, R. Nord, J. Ivers, and R. Little, “Documenting Software Architectures: Views and Beyond”, Pearson Education, 2002.

[15] P. Kruchten, P. Lago, and H. V. Vliet, “Building up and Reasoning about Architectural Knowledge”, in in Proceedings of the Second International Conference on the Quality if Software Architectures (QoSA), Springer-Verlag, 2006, pp. 43-58.

[16] J. Tyree and A. Akerman, “Architecture Decisions: Demystifying Architecture”, in IEEE Softw., vol. 22 (2), 2005, pp. 19-27.

[17] J. S. van der Ven, A. Jansen, J. Nijhuis, and J. Bosch, “Design Decisions: The Bridge between Rationale and Architecture”, Rationale Management in Software Engineering, Springer, 2006, pp. 329-348.

[18] J. van Gurp and J. Bosch, “Design erosion: problems and causes”, in J. Syst. Softw., vol. 61 (2), 2002, pp. 105-119.
[19] P. Kruchten, R. L. Nord, and I. Ozkaya, “Technical Debt: From Metaphor to Theory and Practice”, in IEEE Software, vol. 29 (6), 2012, pp. 18-21.

[20] K. M. Eisenhardt, “Building Theories from Case Study Research”, in Academy of Management Review, vol. 14 (), 1989, pp. 532-550.

[21] J. S. van der Ven, “Making the Right Decision: Supporting Architects with Design Decision Data”, in Software Architecture - 7th European Conference, ECSA, Springer, 2013, pp. 176-183.

[22] S. Thomke, “Enlightened Experimentation: The New Imperative for Innovation (HBR OnPoint Enhanced Edition)”, Harvard Business Review, 2001.

[23] K. M. Eisenhardt and B. N. Tabrizi, “Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry”, in Administrative Science Quarterly, vol. 40 (1), 1995, pp. 84-110.

[24] T. H. Davenport, “How to Design Smart Business Experiments”, in Harvard Business Review, vol. 87 (2), 2009, pp. 68-76.

[25] S. H. Thomke, “Experimentation Matters: Unlocking the Potential of New Technologies for Innovation”, Harvard Business School Press, 2003.

Moonlighting Scrum: An Agile Method for Distributed Teams with Part-Time Developers Working during Non-Overlapping Hours

Philipp Diebold, Constanza Lampasona
Fraunhofer Institute for Experimental Software Engineering
Kaiserslautern, Germany
{philipp.diebold, constanza.lampasona}@iese.fraunhofer.de

Davide Taibi
Software Engineering Research Group
University of Kaiserslautern
Kaiserslautern, Germany
taibi@cs.uni-kl.de

***Abstract*—Scrum and several agile development processes are becoming increasingly popular since they offer the ability to manage volatile requirements. This applies to many types of projects and teams. In case of development teams with moonlight developers working for at most ten non-overlapping hours per week, not all Scrum practices can be applied. In this paper, we introduce Moonlighting Scrum, an adaptation of Scrum aimed at optimizing effectiveness and efficiency by minimizing the amount of communication to the least necessary and maximizing the time invested in development. Our aim is to accomplish this by modifying Scrum practices to achieve a trade-off between development and communication effort to produce the best final results, given the available resources and time. An application of Moonlighting Scrum took place in a real cooperative project and provided interesting results.**

Keywords-agile software development; Scrum; distributed development

I. INTRODUCTION

The adoption of agile software development processes has increased over the years. Agile methodologies are known for being lightweight, which implies moving from heavyweight processes to methods allowing shorter development cycles and more intensive customer involvement. For this reason, many companies have been moving from plan-based development to agile development.

As mentioned by Boehm and Turner [3], these two approaches to software development are considered as opponents. Software development teams often need to stick to one specific process with a defined name and tailor it to their own needs [3]. We believe that the choice of a specific process should be based on tailoring the most advantageous practices from agile and plan-based approaches that best fit the respective team's and project's needs. To answer the issue with which we are confronted, we need such tailoring and a combination of approaches.

We encountered this issue during a software development project we are currently involved in and where requirements evolve over time, which suggested that we use an agile approach. Moreover, our developers are mainly students or researchers doing development in addition to their main activities (study or research), with part-time contracts for at most eleven hours per week without time constraints. Because they are developers in their second job, we call them moonlighters.

In addition to the short time they have available to invest in the project, there is the problem that they are working during different time slots, which makes daily meetings very difficult and pair programming impossible. This suggested the use of a less agile process.

Nonetheless, and although they work part-time, there is still the need for coordinating their work and monitoring project progress.

All these variables in the context of our development projects led us to research the following questions:

RQ1: How much communication is needed to achieve a project's goals? (Effectiveness)

RQ2: How much communication is needed before communication overhead becomes too large? (Efficiency)

Our goal is to find an adequate balance or combination of plan-based and agile approaches which best fits the context of our development projects: distributed moonlighters working during non-overlapping times. The proposed development approach is an adaptation of Scrum, which integrates existing development methods into an agile environment. It addresses a process "to produce best end results, given the current resources and time available" [9, pg. 25]. The approach should be helpful for teams in a similar context because such a constellation is very common in software development, e.g., for open source project or at German universities.

In Section II, we discuss different methodological approaches to software development and their advantages and disadvantages for our development context. In Section III, we introduce an adaptation of a distributed Scrum method that fits our needs, called Moonlighting Scrum. In Section IV, we show how we applied the process in a real project and the measurement plan we applied. Finally, in Section V, conclusions and future work are presented.

II. RELATED WORK

Today's software development processes range from heavy weight plan-based development, such as the waterfall model [15], to incremental and lightweight agile methodologies, such as Extreme Programming [1]. The spiral model combines some aspects of the waterfall model and introduces risk management as a regular step during the process. Unlike the waterfall model, the spiral model iterates through several steps during the entire product development. On the opposite side, agile methodologies include some aspects of iterative models allowing for fast reaction to changes in requirements (Figure 1).

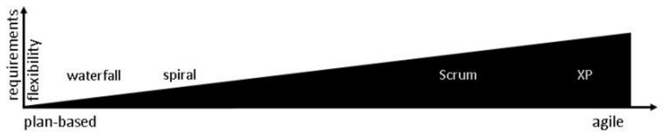


Figure 1. Spectrum of software development processes

Nevertheless, no process fits well to every project context and therefore several other processes have appeared in the literature. In this work, we will introduce the most important approaches we took into account during the design of our model: plan-based development in general and agile development processes such as Scrum, Distributed Scrum, and Extreme Programming.

A. *Plan-based Development*

Plan-driven development approaches (also known as document-centered approaches) such as the waterfall [15], V-Modell XT [7], iterative, or spiral process models [4] are mainly document-centered approaches differing in their execution of the different Software Engineering (SE) phases and have several common requirements on assuring good software quality in their performance. Normally, they are performed for larger projects with larger teams, but with smaller teams the amount of project management stays almost the same [5]. Additionally, plan-based projects try to avoid refactoring because it is very expensive [2], even in a large project, as changes can influence many parts of the product. In contrast to other approaches, plan-based development covers current and future requirements in the architecture. However, this also implies early stable requirements. The developers using such an approach need to work in a plan-oriented manner and have adequate skills or access to external knowledge. The customers of products developed with such plan-based development need to be collaborative, representative, and empowered, since they are mainly involved at the beginning when it comes to making decisions about the requirements.

Plan-based approaches have several disadvantages for the project we want to perform because we only have a small team where all team members are distributed and work during different time slots. In addition, most of the requirements are not stable – and might even not be finished at the beginning of the project. This might lead to a considerable number of refactoring steps, which are expensive in plan-based development.

B. *Scrum*

The vast majority of Scrum practices are not new to SE. Scrum was developed at Easel Corporation in 1993 [1], basically with the same idea behind Barry Boehm's Spiral Model [4].

Scrum speeds up the requirements adaptability of the spiral model with some agile practices from Extreme Programming [13], such as pair programming and daily meetings.

Scrum is a lightweight, iterative, and incremental development model based on three principles: transparency, inspection, and adaptation.

Moreover, Scrum prescribes formal practices for inspection and adaptation:

- Sprint Planning Meeting
- Daily Scrum: daily meeting where each member answers three questions:
    - What did I do yesterday that helped the team meeting the sprint goal?
    - What will I do today to help the team meet the sprint goal?
    - Do I see any impediment that prevents me or the team from meeting the sprint goal?
- Sprint Review
- Sprint Retrospective

Because of the practical requirements, we cannot apply Scrum directly in our team but need to adapt it in a distributed way.

C. *Distributed Scrum*

Distributed teams always face different issues when applying development models. If we increase team distribution, we need to introduce a classification in cooperative SE using globally distributed teams [11]:

- Collocated: Team members are all in the same location.
- Collocated Part-Time: Team members are usually all in the same location but some of them occasionally work off-site. They face similar issues as distributed teams even if they have the opportunity to meet face to face.
- Distributed with Overlapping Work Hours: Team members have a few hours during the workday in which they interact with each other. Scrum meetings can be held during the overlapping time. Sprint planning meetings are more difficult and tend to be less efficient.
- Distributed with No Overlapping Work Hours: Teams have no interaction during their working hours.

In addition to the different levels of distributed teams, we also have to take into account different models that can be considered when using Scrum with distributed teams [17] (Figure 2):

- Isolated Scrums: Teams are isolated across geographies.
- Distributed Scrum of Scrums: Scrum teams are isolated across geographies and integrated by a Scrum of Scrums that meets regularly across geographies.
- Totally integrated Scrums: Scrum teams are cross-functional with members distributed across geographies. Additionally, each team has members in several locations and has its own Scrum Master.

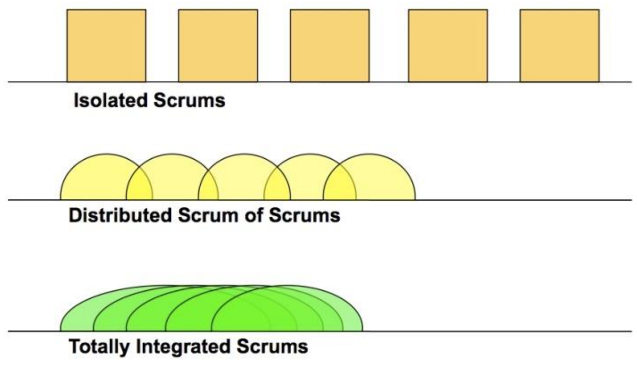


Figure 2. Strategies for distributed Scrum teams [10]

Several works report on the application of Scrum with one of these three categories [8][10][12][13][16].

Sutherland reports on two examples of project management with distributed Scrums of Scrums and fully distributed Scrums [17][18]. These works led to the conclusion that distributed teams can be as productive as small collocated teams if the entire set of teams works as a single team with a global development infrastructure (repository, tracking and reporting tool, and daily meetings). Unlike our work, all teams were composed of several developers working full time and focused on team interaction with daily meetings.

However, not much work has been done to date regarding how to reduce the effort for each team member in teams working during non-overlapping hours.

D. *Extreme Programming*

Extreme Programming [1] is another lightweight software development methodology, which also arose from the need for agility in the development process. Its main idea consists of taking best development practices to the extreme by eliminating anything that might interfere with productivity. The methodology emphasizes incremental development as a response to changing customer needs. Its creator Beck claims that it is especially suitable for small to medium-sized teams. The main practices include pair programming, refactoring, and simple design.

Extreme Programming has been criticized because of its lack of emphasis on design and documentation, which would encourage hacking [9]. It also requires pair programming, which suggests that it might require more effort. People also criticize that it requires constant customer availability and very disciplined teams, which could make its adoption more difficult.

For our context, Extreme Programming is the least suitable methodology, as the team members work only part-time and during different time slots.

TABLE I. COMPARISON OF DEVELOPMENT PROCESSES

	Requirements and SE Practices			**Meetings and Communication**				**Roles**				**Location and Working Hours**				
	Requirements change during project	Refactoring	Pair programming	Daily meeting	Review and retrospective	Planning meeting	Initial meeting	Scrum master	Developers	Product owner/Customer	Project manager	Collocated developers	Collocated part-time developers	Distributed teams, overl. hours	Distributed teams, non-overl. hours	Distributed developers. non-overl. hours
Plan-Based							x		x	x	x	x	x	x	x	x
Moonlighting Scrum	x	x		(x)	x	x	x	x	x	x						x
Scrum	x	x		x	x	x	x	x	x	x		x	x	x	x	
XP	x	x	x	x		x			x	x		x				

III. MOONLIGHTING SCRUM

Distributed teams with part-time developers working during non-overlapping hours are used in several projects. Moreover, at the University of Kaiserslautern, development is often assigned to students with part-time contracts, which requires them to work for a small number of hours per week, in their spare time.

Applying the existing development processes to these teams is always challenging. Table I compares some of the most important development methodologies with Moonlighting Scrum. As we can see from Table I, plan-based development and XP cannot be applied at all, while Scrum has some points in common.

Moonlighting Scrum is a Scrum extension that helps developers to structure the development process with the goal of releasing the best product possible with the available resources in time.

Just like Scrum, Moonlighting Scrum requires sprint planning meetings, sprint reviews, and retrospectives (Figure 3). During the meetings, the whole team and the product owner must meet in person or via video conference.

In Scrum, sprints last from two to three weeks, whereas in Moonlighting Scrum they last from one to two weeks.

Because of the physical distribution and the non-overlapping time for the developers, pair programming cannot be applied and the daily meetings prescribed by Scrum cannot be attended in person.

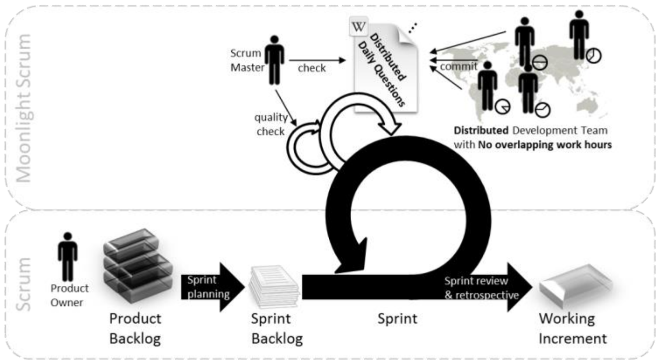


Figure 3. Moonlighting Scrum process schema

Code quality and inspection are the responsibility of the Scrum master, who is in charge of checking overall quality and help the developers preserve a minimum amount of code quality. Moonlighting Scrum is thought to deliver the highest quality possible if limited resources are available.

Therefore, as reported in [12], we substituted morning meetings with an online forum by creating a thread for every six working hours to which each developer posts his/her comments by replying to three questions:

- What have you completed, with respect to the sprint goal, since the last daily meeting?
- What specific tasks, with respect to the sprint goal, do you plan to accomplish until the next daily meeting?
- What obstacles got in the way of completing this work?

The Scrum Master also has to take care of communication efficiency by reducing or increasing the online reporting interval, and is in charge of increasing or decreasing the reporting time based on the team's efficiency.

For this reason, the team members must also answer two additional questions in their online report:

- When did you work (start-end)?
- How much time did you spend on writing this report?

The developers are working for at most ten hours per week and are requested to work for at least two hours continuously. Consequently, the time needed to write the report at the beginning and at the end of their work might take up an important percentage of their working time.

In classical Scrum, daily meetings take 15 minutes. Taking into account 40 working hours per week, daily meetings should take up approximately 3% of the working time.

In contrast, Moonlighting Scrum requires an online report, which usually takes from 5 to 8 minutes, every four to six working hours [12], with at least one report per week. If the developers work for more than six hours per week, they are requested to report twice. The estimated working time used for both cases is 3.5%.

TABLE II. EFFORT REQUIRED

	Hours/ week	Weeks/ sprint	Hours/ meeting	Minutes/ daily report
Scrum	40	2-3	4	15
Moonlighting Scrum	4-10	1-2	2	8

Sprint planning, review, and retrospective meetings in Scrum take four hours per sprint, with sprints lasting from two to three weeks and effort ranging from 3.3% to 5% [12][14].

In Moonlighting Scrum, meetings take suggested two hours with an approximate effort ranging from 6.6% to 12.5% (Table II).

Taking into account the communication issues in a highly distributed team with non-overlapping hours, communication time does not grow significantly, ranging from a maximum of 8% in Scrum to a maximum of 15.5% in Moonlighting Scrum (Table III).

TABLE III. ESTIMATED COMMUNICATION

	Meeting time	Reporting time	Overall time
Scrum	3.3% - 5%	3%	6.3%-8%
Moonlighting Scrum	6.6%-12.5%	3%	9.6%-15.5%

Moonlighting Scrum is applicable to a wide range of projects, from university- and research-based projects to open source projects. In general, the process requires more relative effort for communication than Scrum but allows developing code in a controlled and structured way. The process is applicable whenever we are faced with distributed developers working during non-overlapping hours.

IV. APPLICATION OF MOONLIGHTING SCRUM

Moonlighting Scrum has been applied for the initial development of the software project Technology Repository and Process Configuration Framework [6]. The development started in February 2013 and the first version of the tool was released at the end of May 2013.

A. *Team Organization*

The development team is composed of six people. Some of them are employees of Fraunhofer Institute for Experimental Software Engineering IESE, while the others work for the "Software Engineering Research Group - Processes and Measurement" of the University of Kaiserslautern. All developers have an intermediate level of experience in software development, while none of them has any experience with agile methodologies.

The developers work part-time, with weekly working hours ranging from four to ten, and together spent a total of 39 hours per week on this project.

In order to manage the whole project, from development to communication aspects, we adopted a development infrastructure covering several aspects. The team meets in person during the sprint meeting or, in exceptional cases, via video conference, whereas online reports are recorded in a forum by creating a post for each report.

Sprint retrospectives, planning, and retrospective discussions are led by means of an online integrated tool [19], which allows us to record sprint reports, manage product backlog, and draw burn-down charts.

In addition to this infrastructure and in order to increase collaboration between team members, we also set up a Subversion [13].

B. *Process Measurement and Improvements*

In order to answer the research questions (RQ1 and RQ2), we defined a Goal-Question-Metric measurement plan that allowed us to derive appropriate productivity and communication metrics that impact on effectiveness and efficiency (Table IV).

To define a usable measure for productivity, we considered User Stories (US) as the basic measurement unit.

Since the development is carried out by means of a Rapid Application Development (RAD) Tool (Microsoft Visual Studio 2012), we do not collect code metrics such as lines of code, code complexity, or other metrics because the vast majority of the code is generated automatically by the RAD tool; however, we did define metrics.

Communication time is expressed in terms of time needed to write the online reports and attend the sprint meetings. Total communication time is calculated summing up these two per person. As an example, the training sprint meeting lasted 120 minutes but considering that five people attended the meeting, the total time for the training sprint was 600 minutes (10 hours).

As shown in Table IV, we managed to achieve a sprint meeting duration of two hours or less, except for sprint 1 where the vast majority of topics involved training issues related to the previous training sprint. Total communication time (without reading the online reports) decreased and became stable after two sprints, with effort ranging from 13% to 18%.

On average, communication time required 17% of the total time: 16.4% for the sprint meetings, 0.6% for the online reports, and 83% for development. As a result of this experiment, communication time was slightly higher (17%) than expected (9.6%-15.5%).

TABLE IV. MOONLIGHT SCRUM COLLECTED DATA

	Productivity				**Communication**		
	# days per Sprint	**Tot working hours**	**#Assigned User Stories**	**#Completed User Stories**	**Online report time (minutes)**	**Sprint meeting time (minutes)**	**Communication time/total time (%)**
Training Sprint	10	16	3	3	8	120	63%
Sprint 1	9	50	4	3	26	150	26%
Sprint 2	10	56	6	5	30	120	18%
Sprint 3	14	78	7	6	22	120	14%
Sprint 4	15	84	7	5	27	90	13%
Sprint 5	10	56	5	4	16	120	14%
Sprint 6	11	61	7	5	18	120	17%
Sprint 7	11	61	5	4	26	120	17%
Sprint 8	12	67	3	2	24	120	17%

The application of this process will continue for another three months for project maintenance.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a solution aimed at finding an adequate process for distributed teams with part-time developers working during non-overlapping hours who only have a small amount of effort available per week (ten or less hours per week). Our idea consists of making a trade-off between plan-based and agile development processes. The proposed process is an adaptation of Scrum aimed at optimizing the effectiveness and efficiency of the developers. This means that our goal is to optimize productivity by minimizing the amount of communication to the minimum necessary and maximizing the time invested in development. Our aim is to achieve this with the following instruments:

- Sprint planning, sprint reviews, and retrospective meetings are done in person or via video conference;
- Developers must work for a minimum of two continuous hours;
- Daily meetings are replaced by writing a report in an online forum every six working hours;
- Developers voluntarily report the effort they invest into development and reporting;
- Scrum Master performs code reviews.

The application of Moonlighting Scrum on a real project confirmed that the process can be successfully applied in the university context and helps to keep track of the development steps and to maintain low communication effort.

The project where we applied Moonlight Scrum will continue for another three months for the maintenance phase.

As expected, the process helped us keep track of the development progress. After some initial training and after resolving some technology issues resulting from the new activities required from our developers as well as from the complexity of the domain infrastructure (cyber-physical systems), we were able to maintain low communication overhead.

In future, we will encourage our colleagues working on similar projects to use Moonlighting Scrum process to obtain more evidence to improve it. This should also be done with some open-source development as well as industrial projects to generalize the results.

In addition to this generic aspect we will also try to improve the approach by using other collaboration tools or improving the communication with an online-chat conference system.

ACKNOWLEDGMENT

This paper is based on research being carried out in the ARAMiS (BMBF O1IS11035Ü) project funded by the German Ministry of Education and Research (BMBF).

REFERENCES

[1] K. Beck andC. Andres,“Extreme programming explained. Embrace change” Addison-Wesley Boston, ed. 2, 2005, pp. 64-69.

[2] B. Boehm,“Get ready for agile methods, with care” Computer, vol. 35(1), Jan. 2002, pp. 64-69, doi: 10.1109/2.976920.

[3] B. Boehm and R. Turner,“Balancing agility and discipline. A guide for the perplexed” Addison-Wesley Boston, 2004.

[4] B. Boehm,“A spiral model of software development and enhancement” IEEE Computer, vol. 21(5), May 1988, pp. 61-72, doi:10.1109/2.59 .

[5] M. Ceschi, A. Sillitti, G. Succi, and S. De Panfilis,“Project management in plan-based and agile companies” IEEE Software, vol. 22(3), May-June 2005, pp. 21-27, doi:10.1109/MS.2005.75.

[6] P. Diebold, “How to configure SE development processes context-specifically?”Proc. International Conference on Product-Focused Software Development and Process Improvement (PROFES), Springer LNCS, June 2013, pp. 355-358, doi:10.1007/978-3-642-39259-7_33.

[7] German Federal ministry of the Interios, „V-Model XT. Definition and Documentation on the Web.”Online, http://www.v-model-xt.de, July 2013.

[8] I. Gorton and S. Motwani,“Issues in cooperative software engineering using globally distributed teams” Information and Software Technology, vol. 38(10), Jan. 1996, pp. 647-655, doi:10.1016/0950-5849(96)01099-3.

[9] J. Hunt,“Agile software construction”Springer London, 2005.

[10] J. Kontio, M. Hoglund, J. Ryden, and P. Abrahamsson, “Managing commitments and risks: challenges in distributed agile development” Proc. International Conference on Software Engineering (ICSE), IEEE Press, May 2004, pp.732-733, doi:10.1108/ICSE.2004.1317510.

[11] M. Korkala, and P. Abrahamsson, “Communication in distributed agile development: a case study”Proc. EUROMICRO Conference on Software Engineering and Advanced Applications, IEEE Press,Aug. 2007, pp. 203-210, doi:10.1109/EUROMICRO.2007.23.

[12] L. Lavazza, S. Morasca, D. Taibi, and D. Tosi, “Applying SCRUM in an OSS Development Process: Empirical Evaluation.” Proc. International conference on Agile Software Development (XP), Springer LNBI, June 2010, pp 147-159, doi:10.1007/978-3-642-13054-0_11.

[13] N. Sridhar, M. RadhaKanta, and M. George, “Challenges of migrating to agile methodologies.” Communications of the ACM – Adaptive complex enterprises (CACM), vol. 48, May 2005, pp. 72-78, doi:10.1145/1060710.1060712.

[14] M. Paasivaara, S. Durasiewicz, and C. Lassenius, “Using scrum in a globally distributed project: a case study.” Software Process: Improvement and Practice – Global Software Development, vol. 13(6), Nov. 2008, pp. 527-544, doi:10.1002/spip.v13:6.

[15] W. Royce, “Managing the development of large software systems: concepts and techniques” Proc.Technical Papers of Western Electronic Show and Convention (WESCON),IEEE Press, Aug. 1970, pp. 1-9.

[16] J. Sutherland,“Agile development: lessons learned from the first Scrum”Cutter Agile Project Management Advisory Service: Executive Update, vol. 5, 2004, pp. 1-4.

[17] J. Sutherland, G. Schoonheim, and M. Rijk, “Fully distributed scrum: replicating local productivity and quality with offshore teams.” Proc. Annual Hawaii International Conference on System Sciences (HICSS), IEEE Press, Jan. 2009, pp. 1-8, 2doi:10.1109/HICSS.2009.225.

[18] J. Sutherland, A. Viktorov, J. Blount, and N. Puntikov,“Distributed Scrum: agile project management with outsourced development teams.” Proc. Annual Hawaii International Conference on System Sciences (HICSS), IEEE Press, Jan. 2007, pp. 274a.

[19] RallyDev http://www.rallydev.com (Last access July 2013)

An Agile Maturity Model for Software Development Organizations

Felipe Santana Furtado Soares
UFPE/CIn – Informatics Center – Federal University of Pernambuco
C.E.S.A.R - Recife Center of Advanced Studies and Systems
Recife, Brazil
furtado.fs@gmail.com

Silvio Romero de Lemos Meira
UFPE/CIn – Informatics Center – Federal University of Pernambuco
Recife, Brazil
srlm@cin.ufpe.br

***Abstract*—The transition from traditional methods to agile methods and the changes needed to achieve real benefits from them are difficult to reach. The change affects not only the software development team, but also several areas of an organization and, first and foremost, requires a cultural change. In this context, this paper sets out to define a maturity model that will guide the setting up and running of agile methodologies, based on the Capability Maturity Model Integration (CMMI), in software development organizations. Given the research question considered, the method chosen is a systematic review of the literature, followed by a field study in software development companies. Thus, it is hoped that higher rates of success will be achieved when agile development values, principles and practices are adopted.**

Keywords-Agile metodologies; Maturity Model; Scrum; Lean; CMMI.

I. Introduction

In recent years, substantial transformations have been taking place in the software industry, driven by the demands of the market. Given this backdrop, there has been a demand for organizations to pay special attention to improving their software processes in the pursuit of greater competitiveness and productivity. Therefore, one of the challenges these organizations face is to acquire maturity in their development processes by setting up and running quality models that receive worldwide recognition [1].

At the same time, the market itself imposes deadlines that are more and more competitive and require great agility and high productivity from teams when using processes that bring these about and identifying activities that do not add value to the final product [2].

The challenge then becomes even more complex, as it includes meeting the requirements of a mature model, without spiking productivity, which is based heavily on the control variables of a software development project, while adopting practices of agile processes.

Capability Maturity Model Integration (CMMI) is an approach to improve processes that provides elements that are essential for an effective process. It brings together best practices that address development and maintenance activities, thus covering the entire lifecycle of a product from conception to delivery and maintenance [1].

In the late 90s, several agile methods emerged, including: Adaptive Software Development [34], Crystal [33], Dynamic Systems Development [35], Extreme Programming (XP) [36], Feature Driven Development (FDD) [37] and Scrum. All these methods use agile principles such as iterative cycles, rapid delivery of software that works and simplicity, as defined in the Manifesto for Agile Development [11].

Some authors advocate using the agile approach for managing projects that are conducted in complex environments characterized by many initial uncertainties, and in which there are difficulties in defining the scope and drawing up comprehensive plans, besides a high degree of changes and constant pressures to deliver results within short periods of time. However, the authors claim that the hindsights offered by the traditional project management methods should not be set aside but rather should be combined with the new practices put forward by agile methods [4].

However, some companies still have difficulties in implementing methodologies, either for lack of knowledge, or due to their difficulty in adapting these methodologies to the context of their projects [16]

In this context, after having obtained the correct definition of a maturity model, the expectation is that agile methodologies will be implemented in a systematic and organized way, with more likelihood of their being undertaken successfully. Thus, the main objective of this research is to define a maturity model so as to guide the setting up and running of agile methodologies, based on the CMMI maturity model, in software development organizations, thus resulting in higher success rates when agile development values, principles and practices are adopted.

The paper is divided as follows: Section 2 presents the background overview of CMMI; Section 3 focuses on describing the main agile methodologies and its benefits; Section 4 presents an initial discussion about a maturity model and agile methodologies, showing the difficulties in the transition to agile methods, a technical analysis and an a initial maturity model definition to guide the setting up and running of agile methodologies, based on the CMMI maturity model; The last section concludes this work in progress and presents the next steps.

II. Maturity Models

According to Prado [5], maturity can be defined as "*a way to measure the stage that an organization is at in its ability to manage its projects.*"

The positive and expected results for the company, arising from its growth and maturity, will not come simply from the immediate application of techniques, tools and dissemination of concepts, nor should the best results be expected in the short term. All organizations undergo a maturation process, and this process has to precede excellence. The learning curve for maturity is measured in years [6].

CMMI lays down guidelines to improve the processes of an organization and its ability to manage the development, purchase and maintenance of products and services [3]. The model defines a path towards continuous improvement in terms of five levels of organizational maturity.

The CMMI-based improvement of processes has been accompanied by excellent quantitative results in costs, schedules, productivity, return on investment (ROI), customer satisfaction and product quality [7].

III. Agile Methodologies

In the last ten years, agile methodologies have been gaining ground in the Information Technology and Communication market. Several studies have shown the good results achieved by these companies [12].

A. *The main agile methodologies*

Scrum is a framework for planning and monitoring a project that follows the principles of the Agile Manifesto. Since it is iterative and incremental, it works well in an environment of constant change. It supplies self-managing teams and proposes a form of flexible and adaptable work, not only in relation to the scope and requirements of a project, but also with regard to the exchange of teams, tools, programming languages, etc. [14].

XP is an agile methodology targeted on Software Engineering, and pays greater attention to programming than to management, as the former is the focus of Scrum, which is the reason why these methodologies are normally used together [15]. It was created by Kent Beck in 1996 and seeks to improve a software project by using five essential values: communication, simplicity, feedback, respect and courage.

Large numbers of tools and techniques have been developed to enable organizations to apply Lean concepts and ideas, many of which emerged from TPS (the Toyota Production System), for example, Kanban, JIT (Just in Time), Jidoka, Kaizen, etc. [23].

FDD is an agile methodology for management and software development that combines agile project management practices with a complete approach to object-oriented Software Engineering [24].

B. *Benefits of Agile Software Development*

Cohn [16] consolidated some surveys conducted in 2008 on the benefits of adopting agile software development related to the following matters: cost and productivity, employees' commitment and job satisfaction, time to market, product quality, and stakeholder satisfaction:

- A study conducted by Mah [25] of QSMA has been collecting metrics on productivity and quality for more than 15 years. He conducted a rigorous comparison between 26 agile development projects and a database of 7,500 development projects, mostly traditional ones. The agile projects studied ranged in size from 60 to 1,000 people;
- An extensive survey conducted by Rico [26] on agile projects summarizes 51 studies and academic research papers and gives the main percentage improvements in productivity, cost, quality, scheduling, customer satisfaction and return on investment;
- A survey conducted by the company Version One [13] with more than three thousand people. This is the largest ever survey on the state of adopting agile development. It is international in scope and is the most comprehensive overview of the use of agile development practices;
- A survey conducted by Scott Ambler in February 2008 with 642 people [12];

Regarding the comparison on productivity, research by Mah [25] reports that agile projects are 16% more productive with a confidence level which is statistically significant. These results were corroborated by the research studies below:

- Among the participants in the VersionOne survey [13], 73% found that being agile had improved processes (50%) or had significantly improved them (23%);
- Among the participants in the Ambler survey [12], 82% found that productivity was higher or much higher than before when agile methods were used and only 5% thought that productivity was lower or much lower.

In line with the above research, Rico [26] showed that the average increase in productivity was 88% and the average savings in development costs was 26%.

Regarding the time-to-market, agile teams tend to launch their products faster than traditional teams. VersionOne [13] reported that 64% of participants said that the time-to-market improved (41%) or significantly improved (23%). Mah [25] compared 26 agile projects to the QSMA database which has 7,500 projects and showed that their time-to-market is 37% faster.

IV. A Maturity Model and Agility

Methods, practices and agile techniques for software development promise to increase customer satisfaction [17] by producing higher quality software and accelerating development time [10]. Therefore, organizations that put great effort into improving their processes based on CMMI also believe that agile approaches can supply incremental improvements [20][18].

A. *Difficulties in the transition to agile methods*

The transition to agile methods and the changes necessary to obtain the benefits are difficult to attain. The change affects not only the software development team, but also several areas of the organization; for example, the commercial, marketing and financial areas [16].

Shore [27] and Fowler [28] point out that one of the failures when adopting agile methodologies is related to people: "it is the team that brings success or failure". He also points to the need to use some concepts. For example, for Scrum and XP to be applied together and not just one or the other, given that the former deals with management aspects, while the latter deals with engineering techniques of the product.

Anderson [8] points out that one of the difficulties when adopting these methodologies is associated with the way they are conducted by organizations.

B. Technical Analysis

When a technical analysis is made of models like CMMI and agile methodologies such as Scrum, for example, it is important to note that the perspectives they take are not the same. While maturity models feature a perspective of continuous improvement based on more abstract processes, and aim at meeting the objectives, agile methodologies are more focused on certain contexts and offer a greater level of detail on how to develop a software project.

Maturity models have a broad organizational vision, since they recommend a "path" for continuous improvement, defined in maturity levels. Each level involves various process areas, which include managerial and engineering matters. Conceptually, to be considered as adhering to one of these levels, an organization must meet the goals established for each process area. In addition to the objectives to be met, practices are recommended for each process area that, after having been well performed, immediately lead to goals being achieved.

To reach a CMMI maturity level, the organization must comply with all the process areas of the desired level. CMMI states that "*The only required component of the model is the statement of the specific or generic goal*". This makes it clear that the processes defined do not need to do exactly what is described in typical working products, subpractices and practices. The only requirement is achieving the goals of the process area [1].

C. Agile Maturity Model

Methods, practices and techniques for agile software development promise to increase customer satisfaction [18] by producing higher quality software and accelerating development time [10]. Therefore, organizations that have made a large effort to improve their processes based on CMMI, now also believe that agile approaches can supply incremental improvements [20][18].

Turner [29] comments that, despite the characteristics between agile methods and CMMI being distinct, both have specific plans for software development and pursue what is best so that the organization may produce quality software. Davis [30] reports that despite there being great controversy about the compatibility of Agile Development Methods (ADM) and CMMI, they are not mutually exclusive. He complements this by explaining that there is a place for ADM in CMMI and, more importantly, those who have adopted CMMI may consider adding ADM to their processes.

Paulk [19], lead author of the initial version of the SW-CMM, assessed XP in relation to 18 key process areas of the original SW-CMM. He concluded that XP partially or completely covers 10 of the 13 areas required to achieve Level 3, and is not an obstacle for the other three.

Boehm [17] presented the view that agile and disciplined processes exist on a continuum and can be combined as appropriate based on the risk factors specific to a project.

Jeff Sutherland, co-author of Scrum, reported on a highly productive project and claims that the combination of Scrum and CMMI is more powerful than each of them separately, and he includes guidelines on combining Scrum and CMMI [31].

According to Anderson [9], the way to achieve greater agility with CMMI is to realize that the practices are primarily consultative or indicative, and that to correspond to a CMMI evaluation, an organization must demonstrate that the goals of a process area are being achieved by evidence coming from practices.

In 2008, the Software Engineering Institute (SEI) published a technical report advocating the idea that agile development methods and CMMI best practices are not in disagreement with each other, and that the approaches can be combined successfully [32]. In 2010, the SEI published a book describing case studies that show the integration between CMMI and agile software development [21], but this book did not propose a new maturity model.

According to Marçal [22], it is possible to live peacefully with agile and maturity approaches. Challenges, however, exist and are focused on meeting principles contained in the two approaches. If on the one hand, practices of the maturity model may be added that are not considered in agile methodologies, the essence of these methods should not be unduly shaken. Of course, what the organization should keep in mind is the success of its projects in terms of time, cost and quality. To reach these goals, the flexible and conscious use of maturity models and agile methodologies is valid, provided this is based on an architecture of processes aligned to these goals and the organizational culture.

In this context, the main objective of this study is to define a maturity model so as to guide the setting up and running of agile methodologies, based on the CMMI maturity model, in software development organizations, which result in higher rates of success when agile development values, principles and practices are adopted.

This model is divided into five levels of maturity as follows:

- **Level 1:** initial stage where organizations do not use any methodology and their processes are unpredictable and reactive;
- **Level 2**: the stage where processes are characterized by project. There are processes for planning and monitoring a project, but the organization's vision is by project, i.e., there is no portfolio management of projects. At this level of maturity, setting up agile methodologies starts with Scrum (a focus on managing projects and prioritizing requirements) and a part of the methodology of FDD;
- **Level 3**: the stage where the processes are well defined and characterized by the Organization. There is a standard process with well-defined criteria to

instantiate them at every context of a new project. Engineering processes are implemented with the focus on XP, FDD and Kanban;

- **Level 4**: the stage where the processes are managed quantitatively with the focus on the agile metrics defined in Kanban and FDD;
- **Level 5**: the stage where the process is often optimized, with the focus on continuous improvement of the processes using the principles of Lean Software Development.

V. Conclusion and Future Studies

It was some years ago that agile methodologies became popular in organizations that were seeking environments that conduct software development in a faster and more flexible way. With the purpose of improving the results of software development projects, many organizations choose to introduce agile methods into their development processes. However, many do so in a disorganized way.

This work is part of a proposal for a doctoral thesis and its research methodology is divided into two stages. The first step uses the research instrument called an 'exploratory study', its main objectives being to validate (i) that agile methodologies and maturity models can be used together; (ii) and that there is a need for software development organizations to use a maturity model so as to implement agile methods. This is being undertaken by making a systematic review of the joint use of agile methodologies and maturity models together with a field survey with a view to validating this approach by means of interviews and questionnaires conducted with appropriate staff in some companies that are using agility with the CMMI maturity model.

The second phase will set out to validate the proposal for creating a maturity model for implementing agile methodologies. The main challenge of this validation is related to the possibility of applying the model in a software development company and defining what metrics can be collected before and after adopting the model. Furthermore, the challenge is about isolating the variables before and after measurement to assess whether, in fact, the use of the model contributed to the successful implementation of agile methodologies.

References

[1] CMMI-DEV, CMMI for Development, V1.3 model, CMU/SEI-2010-TR-033, Software Engineering Institute, 2010.

[2] B. Boehm, "A View of 20th and 21st Century Software Engineering", In Proceedings of the 28th international conference on Software engineering (ICSE), ACM, New York, NY, USA, 12-29. DOI=10.1145/1134285.1134288, 2006.

[3] B. Chrissis, M. Konrad, and S. Shrum, "CMMI Guidelines for Process Integration and Product Improvement", Second Edition, Addisson-Wesley, EUA, 2007.

[4] M. Dias, "Um novo enfoque para o gerenciamento de projetos de desenvolvimento de software", São Paulo, 2005. Portuguese Version Only.

[5] D. Prado, "Gerenciamento de Projetos nas Organizações", 2 ed. Belo Horizonte: Editora de Desenvolvimento Gerencial, 2003. Portuguese Version Only.

[6] G. Levin and H. Nutt, "Achieving excellence in business development: the business development capability maturity model", [S.I]: ALLPM, 2005, White paper, Available at: http://www.allpm.com, 10/10/2013.

[7] D. Gibson, R. Dennis, and K. Kost, "Performance Results of CMMI-Based Process Improvement." In: CMU/SEI-2006-TR-004, ESC-TR-2006-004, Pittsburgh, PA: SEI, CMU, 2006.

[8] D. Anderson, "Agile Management": http://www.agilemanagement.net, 10/10/2013.

[9] D. Anderson, "Stretching Agile to fit CMMI Level 3", Agile 2005 Conference, Denver, July, 2005.

[10] D. Anderson, "Agile Management for Software Engineering - Applying the Theory of Constraints for Business Results", Prentice Hall, 2003.

[11] K. Beck, *et al*., "Manifest for Agile Software Development", http://agilemanifesto.org/, 2001, 10/10/2013.

[12] S. Ambler, "Agile adoption rate survey". http://www.ambysoft.com/surveys/, 2008, 2010, 10/10/2013.

[13] VersionOne, "State of Agile Development Survey Results", http://www.versionone.com, 2008, 10/10/2013.

[14] K. Schwaber and M. Beedle, "Agile Software Development with Scrum", New Jersey: Prentice Hall, 2001.

[15] H. Kniberg, "Scrum and XP from the Trenches". Estocolmo: C4Media Inc., 2007.

[16] M. Cohn, "Succeding with Agile: Software Development with Scrum", Bookman, 2011.

[17] B. Boehm and R. Turner, "Management challenges to implementing agile processes in traditional development organizations", IEEE Software, September/October, 2005, pp. 30-39.

[18] B. Boehm and R. Turner, "Balancing Agility and Discipline: A Guide for the Perplexed", AddisonWesley, 2003.

[19] M. Paulk, "Extreme Programming from a CMM perspective". IEEE Software, November, 2001, pp. 19-26.

[20] P. McMahon, "Extending Agile Methods: A Distributed Project and Organizational Improvement Perspective," CrossTalk, The Journal of Defense Software Engineering, vol. 18, issue 5, 2005, pp. 1619.

[21] P. McMahon, "Integrating CMMI and Agile Development: Case Studies and Proven Techniques for Faster Performance Improvement" (SEI Series in Software Engineering), 2010.

[22] A. Marçal, B. Freitas, F. Furtado, T. Maciel, and A. Belchior, "Estendendo o Scrum segundo as Áreas de Processo de Gerenciamento de Projetos do CMMI". CLEI, San José, 2007. Portuguese Version Only.

[23] M. Poppendieck. and T. Poppendieck, "Lean Software Development: An Agile Toolkit", Addison-Wesley, Longman Publishing Co., Inc., Boston, MA, USA., 2003.

[24] J. Luca, "Solutions That Make A Difference", http://www.nebulon.com/articles/index.html, 2008, 10/10/2013.

[25] M. Mah, "How agile projects measure up, and what this means to you". Cutter Consortium Agile Product & Project Management Executive Report 9, 2008.

[26] D. Rico, "What is the ROI of agile vs. traditional methods? An analysis of extreme programming, test-driven development, pair programming and Scrum", 2008.

[27] J. Shore and S. Warden, "The Art of Agile Development". O'Reilly Media, Inc., Sebastopol, CA, EUA, 2008.

[28] M. Fowler, "FlaccidScrum". http://www.martinfowler.com, 2009.

[29] R. Turner and A. Jain, "Agile Meets CMMI: Culture Clash or Common Cause," In proceedings of the Second XP Universe and First Agile Universe Conference on Extreme Programming and Agile Methods XP/Agile Universe, 2002, pp. 153-165.

[30] C. Davis, M. Glover, J. Manzo, and D. Opperthause, "An Agile Approach to Achieving CMMI", http://www.agiletek.com, March 2007, 10/10/2013.

[31] J. Sutherland, C. Ruseng, and K. Johnson., “Scrum and CMMI Level 5: The Magic Potion for Code Warriors”, The 12th annual European Systems and Software Engineering Process Group Conference European SEPG 2007 11-14th June, Amsterdam, 2007.

[32] H. Glazer, J. Dalton, D. Anderson, M. Konrad, and S. Shrum, “CMMI or Agile: Why Not Embrace Both”. CMU/SEI-2008-TN-003, November, 2008.

[33] A. Cockburn, “Crystal Clear: A Human-Powered Methodology for Small Teams”, 1st edition, Addison-Wesley Professional, 2005.

[34] J. A. Highsmith, "Adaptive Software Development: A Collaborative Approach to Managing Complex Systems", New York, NY, Dorset House Publishing, 2000.

[35] J. Stapleton, "DSDM: Dynamic Systems Development Method: The Method in Practice", Addison-Wesley, Boston, MA, 1997.

[36] K. Beck, "Extreme Programming Explained: Embrace Change", 2nd Edition, Boston, MA, Addison-Wesley Professional, 2005.

[37] S. R. Palmer and J. M. Felsing, "A Practical Guide to Feature-Driven Development", NJ, Prentice Hall, 2002.

Using the Analytical Hierarchy Process as a Ranking Tool for User Story Prioritization Techniques

Sultan Alshehri and Luigi Benedicenti
Software Systems Engineering
University of Regina, Regina
Regina, SK, Canada
Email: aljumais@uregina.ca, luigi.benedicenti@uregina.ca

Abstract— **The Analytic Hierarchy Process (AHP) has been applied in many fields and especially to complex engineering problems and applications. AHP is capable of structuring decision problems and finding mathematically determined judgments built on knowledge and experience. This suggests that AHP should prove useful in agile software development, where complex decisions occur routinely. This paper provides a ranking approach to help stakeholders select the best prioritization technique for prioritizing the user stories. A case study demonstrated the effectiveness of this approach.**

Keywords-Extreme Programming; User Stories; Analytic Hierarchy Process.

I. INTRODUCTION

The quality of Extreme Programming (XP) development results from taking 12 core practices to their logical extremes [1]. One such practice is the planning game, in which customers and developers cooperate to develop requirements that produce the highest value for customers as rapidly as possible. This is accomplished as follows. Customers write system requirements as user stories. User stories are defined as “short descriptions of functionality told from the perspective of a user that are valuable to either a user of the software or the customer of the software” [2]. Developers review the stories to ensure domain-specific information is sufficient for their implementation. Developers evaluate user stories using story points to identify the complexity and cost of their implementation. Then, user stories are broken down into small tasks. Finally, customers and developers collaborate in prioritizing user stories based on their value and other relevant factors.

To reconcile conflicting opinions among them, customers and developers often adopt a prioritization technique [3,4,5]; but, this adoption process is usually not formalized. In this paper, the Analytical Hierarchy Process (AHP) is utilized as a well-structured multi-criteria decision making tool to help XP software development teams rank six prioritization techniques: 100-Dollar Test (Cumulative Voting), MoSCow, Top-Ten Requirements, Kano Model, Theme Screening, Relative Weighting.

This paper is organized as follows: Sections 2 to 6 describe the AHP method; the six prioritization techniques are presented in Section 7; four criteria for ranking the prioritization techniques are proposed in Section 8; a case study, its results and its findings are presented in Section 9 and 10, and Section 11 concludes the paper.

II. RELATED WORK

There is no consensus in the literature on the most important factors determining the priority of system requirements. However, almost all the factors taken into consideration aim to maximize the value delivered to the customer. Bakalova et al. proposed to use project context, effort estimation, dependencies, input from the developers, learning experiences and external change [6]. Hoff et al. relied on four factors: cost-benefit to the organization, impact of maintenance, complexity and performance effects [7]. They also considered fixed errors, requirement dependencies, complexity, and delivery data/schedule as ancillary factors. Somerville and Sawyer prioritize requirements based on the viewpoint approach that represents information about the system requirements from different perspectives representing different types of stakeholder [8]. Davis used Triage as an evaluation process considering time, available resources, and requirements interdependencies [9]. Lutowski prioritized the requirements based on the importance or immediacy of need [10]. Bhoem considered the cost of implementing the requirement as the most important factor for prioritization [11]. In Bhoem’s work, cost is related to the technical environment, complexity, quality, timeframe, documentation, availability reusable software, participant competencies, and stability of requirements. Berander and Andrews surveyed the literature and found common aspects in prioritizing requirements such as penalty, cost, time, risk and volatility [12]. The authors added that other aspects like financial benefits, competitors, release theme, strategic benefit, competence/resource, and ability to sell should also be considered.

In the agile methodology domain, Patel and Ramachandran prioritized user stories based on business functionality, customer priority, core value, market values, implementation cost, and business risk [13]. Many well-established prioritization technique available are applicable to requirements prioritization: Ping Pong Ball, Pair-Wise Analysis, Weighted Criteria Analysis, Dot Voting, Binary Search Tree, Ranking, Numeral Assignment Technique, Requirements Triage, Wieger's Matrix Approach, Quality Function Deployment, Bucket technique, Cumulative Voting, Round-the-Group Prioritization, Theory-W, and Theme Scoring [14,15].

Changes to requirements in a plan-based environment are difficult and costly. Thus, a change the user considers simple may translate into a painful process for the developers. By definition, this is not the case for requirements in agile methods. This fundamental difference may have an impact on the optimal choice of prioritization technique.

Mead conducted a case study to determine the most suitable requirements prioritization methods to be used in software development [5]. This study compared three common methods: Numeral Assignment Technique, Theory-W, and AHP. The prioritization method comparison was based on five aspects: clear-cut steps, quantitative measurement, high maturity, low labor-intensity, and shallow learning curve. The results indicated that the AHP ranked the highest score of 16, while the Numeral Assignment Technique scored a 12, and Theory-W scored an 8.

III. Methodology

The primary objective of this study is to investigate how the AHP can be used to rank the user stories prioritization techniques. The methodology used in this study is the case study methodology described in [16].

The following research questions provided a focus for our case study investigation:

(1) How does the AHP help select a prioritization technique for user stories?

(2) How do the AHP results affect the relationships among developers relation and their performance?

The units of analysis for this study derive from these research questions. The main focus is to rank several tools that can be used to prioritize user stories. Accordingly, ranking and the evaluation process are two the units of analysis for this study. Also, we consider the developers view of how the AHP benefits each XP practice. As result, our study is designed as multiple cases (embedded) with two units of analysis.

IV. Data Collection and Sources

In the beginning of the study, we found the criteria affecting the ranking process and helping to examine the AHP tool ability and benefits. This data was collected from literature review and previous studies. To increase the validity of this study, data triangulation was employed. The data sources in this study were:

1. Archival records such as study plans from the graduate students.
2. Questionnaire given to the participants when developing the XP project.
3. Open-ended interviews with the participants.
4. Feedback from the customer.

V. Case Study

The case study was conducted in the Advanced Software Design course offered to graduate students in Fall 2012 at the University of Regina. The participants were 12 Master's students and a client from a local company in Regina. Participants have various levels of programming experience and a good familiarity with XP and its practices. The students background related to the case study included several programming languages such as Java, C, C#, and ASP.net. All participants had previous project development experience. The study was carried out throughout 15 weeks; the students were divided into two teams. Both teams were assigned to build a project called "Issue Tracking System" brought in by the client along with a set of requirements compatible with current industry needs. The project evolved through 5 main iterations and by the end of the semester, all software requirements were implemented. The students were requested to try all requirements in each prioritization technique before applying AHP to rank them. Participants were given detailed lectures and supporting study materials on Extreme Programming practices that focused on planning game activities which included writing user stories, prioritizing the stories, estimating process parameters, and demonstrating developers commitments. The students were not new to the concept of XP, but they gained more knowledge and foundation specifically in the iteration plan, release planning and prioritizing the user stories. In addition, the students were exposed to the AHP methodology and learned the processes necessary to conduct the pairwise comparisons and to do the calculations. Several papers and different materials about AHP and user stories were given to the students to train them and increase their skills in implementing the methodology. Finally, a survey was distributed among students to get further information about their personal experiences and knowledge.

VI. The Analytical Hierarchy Process

AHP is a systematic approach for decision-making that involves the consideration of multiple criteria by structuring them in a hierarchical model. AHP reflects human thinking by grouping the elements of a problem requiring complex and multi-aspect decisions [17]. The approach was developed by Thomas Saaty as a means of finding an effective and powerful methodology that can deal with complex decision-making problems [8]. AHP comprises the following steps: 1) Structure the hierarchy model for the problem by breaking it down into a hierarchy of interrelated decision elements. 2) Define the criteria or factors and construct a pairwise comparison matrix for them; each criterion on the same level of the decision hierarchy is compared with other criteria in respect of their importance to the main goal. 3) Construct a pairwise comparison matrix for alternatives with respect to each objective in separate matrices. 4) Check the consistency of the judgment errors by calculating the consistency ratio. 5) Calculate the weighted average rating for each decision alternative and choose the one with the highest score. More details on the method, including a step-by-step example calculation, are found in [17].

Saaty developed a numerical scale for assigning the weight for criteria or alternative by giving a value between 1 (equal importance) and 9 (extreme importance) [18]; see Table 1 for details.

TABLE 1. AHP NUMERICAL SCALE DEVELOPED BY SAATY..

Scale	Numerical Rating	Reciprocal
Equal importance	1	1
Moderate importance of one over other	3	1/3
Very strong or demonstrated importance	7	1/7
Extreme importance	9	1/9
Intermediate values	2,4,6,8	1/2, 1/4, 1/6, 1/8

VII. PRIORITIZATION TECHNIQUES

There are several methods for prioritizing the system requirements; the six most commonly used can be summarized as follows:

1) *The 100-Dollar Test (Cumulative Voting)*

This is a straightforward technique described by Leffingwell and Widrig where each stakeholder gets 100 imaginary units (money, hours, etc) to distribute among the given requirements [19]. If the requirements are too many, it is recommended to use more units of value for more freedom in the prioritization [20]. After distributing the units on the requirements, stakeholders calculate the total for each requirement and rank the requirements accordingly.

2) *MoSCoW*

This is one of the methods for prioritization originating from the Dynamic Software Development Method (DSDM) [21]. The requirements are classified into four groups depending on the importance of the functional requirements [22]:

- M: MUST have this. It is the highest priority and without it the project considered a failure.
- S: SHOULD have this requirement if possible. Customer satisfaction depends on this requirement. But we cannot say its absence causes a project to fail.
- C: COULD have this requirement if it doesn't affect anything else.
- W: WON'T have the requirement this time but WOULD like to in the future.

This technique helps understand customer needs. The problem with this method is the difficulty of distinguishing the terms "Must" and "Should" as they both express a customer preference or desire..

3) *Top-Ten Requirements*

In this approach, the stakeholders select their top ten requirements without giving them a specific priority [23]. This is to avoid the conflict between stakeholders that may arise from the desire to support specific requirements. However, if stakeholder alignment is low, it is possible that none of the choices for some stakeholders will appear in the aggregated top priority requirement list.

4) *Kano Model*

This method was established for product development by Noriako Kano in 1987 to classify the requirements into five categories based on the answers to two questions about every requirement: 1) functional question: "How do you feel if this feature is present?"; 2) dysfunctional question: "How do you feel if this feature is NOT present?" [24].

The customer has to choose one of the five possible options for the answers [25]:

1. I like it.
2. I expect it.
3. I'm neutral.
4. I can tolerate it.
5. I dislike it.

5) *Themes Screening*

This is a technique employed when stakeholders have many relevant user stories that need to be grouped together. While writing the stories, stakeholders eliminate similar stories or ones that have already been covered by others. Then they follow the steps below [26]:

1. Identify 5-9 (approximately) selection criteria that are important in prioritizing the themes.
2. Identify a baseline that is approved and understood by all the team members.
3. Compare each theme to the baseline theme for each criterion. Use "+" for themes that rank "better than" the baseline theme, "-" for themes that rank "worse than" the baseline theme and "0" for themes that rank "equal" to the baseline theme.
4. Calculate the "Net Score" by summing up all the plusses and minuses. Rank as number one the theme that received the highest Net Score.

6) *Relative Weighting*

This technique involves the evaluation of each requirement based on the effect of its presence and its absence. A scale from 0 to 9 is identified for each requirement, 0 being a low effect and 9 being a high effect. Stakeholders will give every feature a value for its presence as well as a penalty for its absence and estimate its implementation cost. The priority is calculated by dividing the total value by the total cost to generate a prioritization indicator [26].

VIII. PROPOSED CRITERIA FOR RANKING

To rank each technique, it is necessary to determine the most important criteria that affect the participants when choosing a prioritization process. The resulting criteria will be compared among each other. Finally, the prioritization techniques will be compared against each of the criteria [27]. In this paper, we propose four prioritization criteria that emerged during the course of the case study we conducted, but the method described in this paper can be applied to any set of criteria. The criteria shown below are simply illustrative of the prioritization method.

1. Simplicity: What is the simplest prioritization technique in terms of ease of understanding and application?
2. Time: Which one of these techniques will save the most time when the team applies it to the user stories?
3. Accuracy: Which one of these techniques will give the most accurate results?
4. Collaboration: Which one of these techniques will achieve the highest degree of collaboration among the stakeholders and the XP team in general?

IX. AHP IN PRACTICE

The first step in the Analytic Hierarchy Process is to structure the problem as a hierarchy. In this paper, such a hierarchy includes three levels. The top level is the main objective: ranking the prioritization techniques. The second level is the prioritization criteria: simplicity, time, accuracy, and collaboration. The third level is the alternatives: 100-Dollar, Top-Ten, Kano Model, Theme Screening, Relative Weighting, and MoSCow. Fig. 1 illustrates the AHP hierarchy we chose for this paper.

Then, the hierarchy is used to generate appropriate AHP tables. All team members receive these tables, which shortens the time to fill them and facilitates the comparison process. A cover page dedicated to collecting general information of each team member including experience, type, and level of programming skills is also handed out. A matrix is then used to compare the four prioritization criteria.

Accordingly, we required all students to use the prioritization techniques throughout the project to experience their advantages and disadvantages. Then, we asked the students to evaluate these techniques based on the prioritization criteria. To accomplish this, we provided them with the AHP tables and cover page described above.

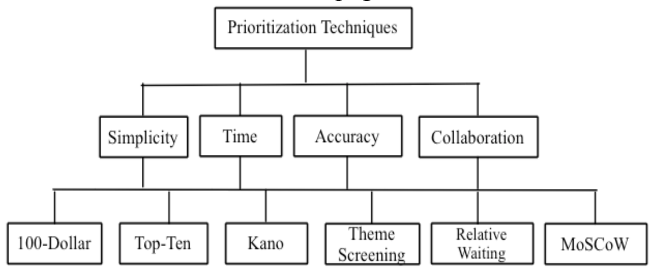


Figure 1. AHP Structure for Ranking the Prioritization Techniques

The students first compared the criteria among each other using the Saaty scale, ranging from 1 to 9. The students used a checklist with the following questions:

- Which is more important: simplicity or time and by how much?
- Which is more important: simplicity or accuracy and by how much?
- Which is more important: simplicity or collaboration and by how much?
- Which is more important: time or accuracy and by how much?
- Which is more important: time or collaboration and by how much?
- Which is more important: accuracy or collaboration and by how much?

After finishing the criteria comparisons, the students had to evaluate all the prioritization techniques against each other based on each criterion every time. An example follows:

- In term of simplicity, which is simplest: 100-Dollar or Top-Ten and by how much?

The same questions and comparisons were repeated for all prioritization techniques and criteria.

X. FINDINGS AND RESULTS

Each student individually evaluated the prioritization techniques based on the criteria mentioned earlier. The Expert Choice software [28] was used to calculate the aggregation results for the entire two teams.

The results for Team 1 show that the highest rank was given to the relative weighting technique, followed by MoScoW, Theme Screening, Kano, Top-Ten and 100-Dollar. Table 2 provides the relative scores of each ranking as percentages.

The software also allows us to examine the importance of each criterion as perceived by Team 1 (Fig. 2). It appears that accuracy was the most relevant criterion for the team, followed by simplicity, collaboration and time.

TABLE 2. PRIORITIZATION TECHNIQUE RANKING FOR TEAM 1

Technique	Scores
Relative Weighting	24.39%
MoScoW	20.38%
Them Screening	17.70%
Kano	15.81%
Top-Ten	12.75%
100-Dollar	8.97%

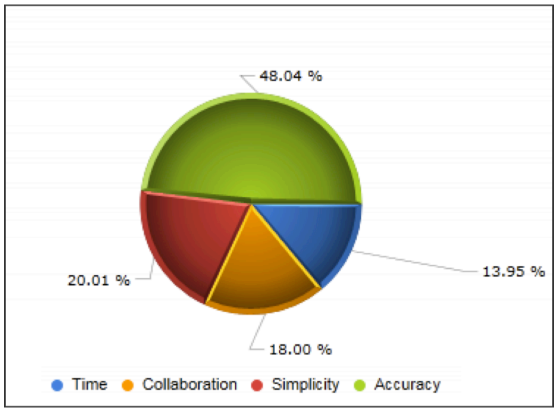


Fig.2 The Importance of the Criteria by Team 1

TABLE 3. PRIORITIZATION TECHNIQUE RANKING FOR TEAM 2

Technique	Scores
Relative Weighting	32.67 %
Top-Ten	26.12 %
MoScoW	15.44 %
Theme Screening	15.35 %
100-Dollar	7.15 %
Kano	3.27 %

The results for Team 2 paint a somewhat different picture: the Relative Weighting technique is still on top, but it is followed by Top-Ten, MoScoW, Theme Screening, 100-Dollar and finally Kano. Table 3 provides the relative scores of each ranking as percentages.

As for the importance of each criterion as perceived by Team 1 (Fig. 3), it appears that accuracy was still the most relevant prioritization criterion, followed by time, collaboration and simplicity.

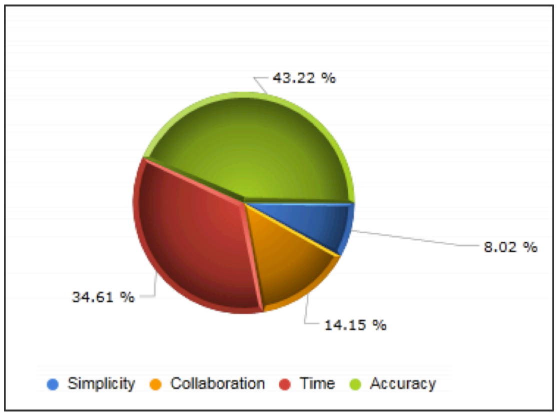


Fig. 3 The Importance of the Criteria by Team 2

XI. OBSERVATIONS

a) AHP Ranking Result

- When all the criteria were considered together, the Relative Weighting technique was ranked the highest by both teams. The MoScoW technique was ranked in the second position by Team 1 and third position by Team 2. The 100-Dollar technique was ranked in the last position by Team 1 and in the second to last position by Team 2.
- Both teams considered accuracy as the most important criteria. Simplicity in Team 1 and time in Team 2 respectively were considered to be the second highest important criterion.
- When the prioritization techniques were ranked considering each criterion individually, we found that for Team1 the MoScoW technique was ranked the highest in terms of simplicity and time criteria. Relative weighting was ranked the highest in terms of accuracy and collaboration criteria. Results related to Team2 are slightly different: the Top-Ten technique ranked the highest in terms of simplicity and time criteria. Relative weighting ranked the highest in terms of accuracy and collaboration criteria.
- These results are indicative of different choices made in each team. Although the ranking was achieved through individual comparisons, the group behavior was consistent as reflected in the consistency scores, which allowed the software to aggregate results from team members.

b) Interview Results

The interview was conducted after showing the participants the results of the AHP evaluation for all the XP practices. Some of the results were surprising and others were expected. The interview included open questions to obtain the students' general opinions about AHP, the advantages and disadvantage of the using AHP, and the best experience of AHP among all the XP practices. As noted previously, the data was collected in the form of handwritten notes during the interviews. These notes were organized in a folder for the sake of easy access and analysis.

From the interviews, we found very positive feedback from the participants regarding AHP. It was felt that AHP resolved any conflicting opinions and brought each team member's voice to the decision in a practical way. AHP also emphasized the courage of the team by letting every opinion be heard. The time and the number of comparisons were the main concerns of the participants. All of them recommended using AHP in the future with XP. There were a few additional recommendations as well, such as developing an automated tool to reduce the time required for the AHP calculation, adding the mobility features, performing cost and risk analysis, and trying AHP in other XP areas and studying the outcomes.

c) Questionnaires

Questionnaires were also given to the participants in order to obtain their perceptions of and experiences with AHP. The questionnaires were divided into two main parts. The first part contained questions about AHP as a decision and ranking tool. The second part contained questions regarding the direct benefits of the XP practice and investigated the participants' satisfaction. We used a seven-point Likert scale to reflect the level of acceptability of the AHP tool as follows:

1. Totally unacceptable
2. Unacceptable.
3. Slightly unacceptable.
4. Neutral.
5. Slightly acceptable.
6. Acceptable.
7. Perfectly Acceptable.

Once the participants completed the questionnaire, we aggregated the responses and presented the total percentage of the acceptability for each statement.

The total percentage of the acceptability was calculated as follows:

d) The total percentage of acceptability (TPA)

= The average of the score for each team * 100 / 7.

e) The average of the score for each team =

= The sum of the scores given by the team members / number of the team.

The following percentages show the acceptability level for the AHP as a ranking tool:

- Improving team communication: Team 1 scored 83% and Team 2 scored 86%.
- Creating a healthy discussion and learning opportunities: Team 1 scored 74% and Team 2 scored 93%.
- Clarifying the ranking problem: Team 1 scored 86% and Team 2 scored 93%.
- Resolving conflicting opinions among members: Team 1 scored 78% and Team 2 scored 93%.
- Increasing team performance: Team 1 scored 74% and Team 2 scored 88%.

XII. VALIDITY

Construct validity, Internal Validity, External Validity and Reliability describe common threats to the validity of the study [29]. "Empirical studies in general and case studies in particular are prone to biases and validity threats that make it difficult to control the quality of the study to generalize its results" [30]. In this section, relevant validity threats are described. A number of possible threats to the validity of this work can be identified.

a) Construct validity

Construct validity deals with the correct operational measures for the concept being studied and researched. The major threat to this study is the small number of participants in each case study.

This threat was mitigated by using several techniques in order to ensure the validity of the findings.

- Data triangulation: A major strength of case studies is the possibility of using many different sources of evidence [29]. This issue has been taken into account through the use of surveys and interviews with different types of participants from different environments with various levels of skills and experiences, and through the use of several observations as well as feedback from those involved in the study. By establishing a chain of evidence, we were able to reach a valid conclusion.
- Methodological triangulation: The research methods employed were a combination of a project conducted to serve this purpose, interviews, surveys, AHP results comparisons, and researchers' notes and observations.
- Member checking: Presenting the results to the people involved in the study is always recommended, especially for qualitative research. This is has been done by showing the final results to all participants to ensure the accuracy of what was stated and to guard against researcher bias.

b) Internal validity

Internal validity is only a concern for an explanatory case study [29], and it focused on establishing a causal relationship between Students and educational restraints.

This issue can be addressed by relating the research questions to the study's propositions and other data sources providing information regarding the questions.

c) External validity

External validity is related to the domain of the study and the possibilities of generalizing the results. To provide external validity to this study, we will need to conduct an additional case study in the industry involving experts and developers and then observe the similarities and the differences in the findings of both studies. Thus, future work will contribute to accrue external validity.

d) Reliability

Reliability deals with the data collection procedure and results. Other researchers should arrive at the same case study findings and conclusions if they follow the same procedure. We address this by making the research questions, case study set up, data collection and analysis procedure plan available for use by other researchers.

XIII. CONCLUSIONS

After using AHP to rank the common requirement prioritization techniques used in XP development to prioritize the user stories, AHP was found to be a relevant and useful tool that affords very good vision to stakeholders when they want to decide on which prioritization technique is the most suitable. Considering simplicity, time, accuracy and collaboration when selecting a prioritization technique could bring many advantages to the XP team, including the stakeholders. The relative weighting technique was the most preferred method for both teams in our case study, but the procedure we followed is general and thus the ranking can change depending on the team. More importantly, though, AHP helped students evaluate each prioritization technique from different viewpoints. In addition, they could mathematically reconcile the conflict of opinions among them. AHP introduces a cooperative decision making environment, which accelerates the XP development process and maximizes the effectiveness of the software developed.

REFERENCES

[1] K.Beck, "Extreme Programming Explained: Embrace Change," 2nd edition, Addison Wesley, 2000.

[2] M.Cohn. "Advantage of User Stories for Requirements, Information Network," (October 2004)

[3] K.Wiegers, "Software Requirements," Microsoft Press, Redmond, In Engineering and Managing Software Requirements,2003.

[4] Lawson. "Software Requirements-Styles and Techniques," Pearson Education, Essex, 2002.

[5] R. Mead, "Requirements Prioritization Introduction," Software Engineering Institute, 2006-2008 Carnegie Mellon University.

[6] Z. Bakalova, M. Daneva, A. Herrmann, and R. Wieringa, "Agile Requirements Prioritization: What Happens in Practice and What Is Described in Literature," In D. Berry & X. Franch (Eds.), Requirements engineering: Foundation for software quality, LNCS, vol. 6606, 2011, pp. 181-195. Heidelberg, Germany: Springer Berlin Heidelberg.

[7] G. Hoff, A. Fruhling, and K.Ward, "Requirements Prioritization Decision Factors for Agile Development Environments," University of Nebraska at Omaha, 2008.

[8] I. Sommerville and P. Sawyer, "Requirements Engineering: A Good Practice Guide," John Wiley & Sons Ltd, Chichester, England, 1997.

[9] A. Davis, "The Art of Requirements Triage," IEEE Computer, Vol. 36, No. 3, March 2003, pp. 42- 49.

[10] R. Lutowski, "Software Requirements," Auerbach Publications, Boca Raton, 2005.

[11] B. Boehm, "The High Cost of Software," Practical Strategies for Developing Large Software Systems, Addison- Wesley, Reading MA, 1975.

[12] P. Berander and A. Andrews, "Requirement Prioritization," in Engineering and Managing Software Requirements, Berlin, Deutschland, 2005.

[13] C. Patel and M. Ramachandran, "Story Card Based Agile Software Development," in International Journal of Hybrid Information Technology, vol. 2, no. 2, April.2009.

[14] Z. Racheva, M. Daneva, and L. Buglione, "Supporting the Dynamic Reprioritization of Requirements in Agile Development of Software Products," Second International Workshop on Software Product Management, 2008.

[15] Q. Ma, "The Effectiveness of Requirements Prioritization Techniques for a Medium to Large Number of Requirements: A Systematic Literature Review," thesis for a degree of master of Computer and Information Sciences, Auckland University of Technology, 2009.

[16] K. Yin, "Case Study Research: Design and Methods," Second Edition, SAGE Publications, 1994.

[17] N. Tiwari. "Using the Analytic Hierarchy Process (AHP) to Identify Performance Scenarios for Enterprise Application" (2006)

[18] T. Saaty, "The Analytic Hierarchy Process," McGraw-Hill, New York, 1980.

[19] D. Leffingwell and D. Widrig, "Managing Software Requirements: A Use Case Approach," 2nd ed. Addison-Wesley, Boston (2003).

[20] P. Berander and C. Wohlin, "Different in Views between Development Roles in Software Process Improvement – A Quantitative Comparison," In: Proceedings of the 8th International Conference on Empirical Assessment in Software Engineering (EASE 2004). IEE, Stevenage, 2004, pp. 57-66.

[21] K. Waters, "Prioritization Using MoSCoW," Agile Planning, (12 January 2009)

[22] The MoSCoW Prioritization Technique, LMR Technologies, Agile Practices: Scrum, XP, Lean, Kanban: www.lmrtechnologies.com [retrieved: October, 2013].

[23] K.Wiegers, First Things First: Prioritizing Requirements, Software Development, vol. 7, no. 9, September 1999.

[24] E. Zultner, H. Mazur. The Kano Model: Recent Developments, Richard QFD Institute, Austin, Texas,2006.

[25] A. Hand, "Applying the Kano Model to User Experience Design," UPA Boston Mini-Conference,May 2004.

[26] M. Cohn, "User Stories Applied for Agile Software Development," Addison-Wesley Professional; 1 edition (March 11, 2004)

[27] T. Saaty, "How to Make a Decision: the Analytic Hierarchy Process," Interfaces, vol. 24, no. 6, 1994, pp.19-43.

[28] Expertchoice for Collaborative Decision Making: http://www.expertchoice.com [retrieved: October, 2013].

[29] R.K. Yin, Case Study Research – Design and Methods, 3rd edition, Sage Publications, London, 2003.

[30].R. Lincke, "How do PhD Students Plan and Follow-up their Work? – A Case Study," School of Mathematics and Systems Engineering, University Sweden.

Expert Estimation and Historical Data: An Empirical Study

Gabriela Robiolo
Universidad Austral
Av. Juan de Garay 125
Buenos Aires, Argentina
grobiolo@austral.edu.ar

Silvana Santos
Universidad Nacional de La Plata
Calle 50 y 20
La Plata, Argentina
silvanasantos@gmail.com

Bibiana Rossi
Univ. Argentina de la Empresa
Lima 717
Buenos Aires, Argentina
birossi@uade.edu.ar

***Abstract*— Expert estimation is the estimation strategy which is most frequently applied to software projects; however, this method is not very much reliable as the accuracy of the estimations thus obtained is always influenced by the level of experience of the expert. As part of the experts' experience is made up by the information they obtain from historical data, we wanted to learn about the value such historical data has for an expert estimator. To do so, we designed an empirical study. We compared the accuracy of the estimations made with several estimation methods based on productivity, size, and analogies which use historical data, to that obtained with expert estimation. We used two similar applications; one was used as the target application and the other one was used to obtain historical data. The results show that the accuracy of expert estimation is affected by the expert's work experience, the level of experience he/she has in the technologies to be used to develop the applications, and his/her level of experience in the domain of the applications. The use of historical data may improve the intuitive expert estimation method when the work experience, the experience in the technologies to be used to develop the application, and the experience in a given domain is low, as well as when the team velocity is unknown.**

Keywords—Expert; Expert Estimation; Effort Estimation; Empirical Study; Historical Data.

I. Introduction

Expert estimation is the estimation strategy which is most frequently applied today to estimate the effort involved in the development of software projects, and this is so because there is evidence in favor of using it [1]. However, the estimations thus obtained are far from being as accurate as we would like them to be so, if we expect to improve estimation accuracy, further research should be carried out in order to understand how the estimation process works.

With this goal in mind, we found out that the compilation of information about cost estimation made by Jørgensen and Shepperd [2] in 2004 is extremely valuable, since they systematically reviewed papers already written on cost estimation studies and they provided recommendations for future research. They found out that there are few researchers working in this field and that there is no adequate framework to develop high quality research projects that may lead to concluding evidence. Consequently, they suggested the following improvements in the field of research: (a) deepen the study of the basic aspects of software estimation, (b) widen the research on the current, most commonly used estimation methods in the software industry, (c) perform studies that support the estimation method based on expert judgment, instead of replacing it with other estimation methods and (d) apply cost estimation methods to real situations.

As we completely agree with their diagnosis, we believe research on expert estimation has become mandatory, if more accurate estimations are to be obtained.

As far as we know, expert estimation may be said to be based on both intuition, which is acquired by the developer through his daily work experience, and analogy. In fact, such analogy will be made by using both the information the estimator has in his memory and the historical data he may obtain [2]. Although all experts are expected to have some experience, the types of experience they have may be very different, and their estimation performances will surely be different too. Besides, even in cases in which the expert is supposed to have wide experience, there will be factors that will undoubtedly affect his estimations. For example, the domain where the software estimation must be made could be new to him, the team he would work with may have been recently created or the technological environment may not have been previously used.

In Agile contexts, in particular, there is another critical aspect to be dealt with: not knowing the velocity at which the developing team works. Actually, Cohn [3] suggested that one of the challenges when planning a release is estimating the velocity of the team. He mentioned three possible ways to estimate velocity. Firstly, estimators may use historical averages, if available. However, before using historical averages, they should consider whether there have been significant changes in the team, the nature of the present project, the technology to be used, and so on. Secondly, estimators may choose to delay estimating velocity until they have run a few iterations. Cohn thinks that this is usually the best option. Thirdly, estimators may forecast velocity by breaking a few stories into tasks and calculating how many stories will fit into the iteration.

Bearing in mind the present working conditions, as described in the two previous paragraphs, and in order to deepen our knowledge about expert estimation, as recommended by Jørgensen and Shepperd [2], we decided to research on the importance of historical data when performing expert estimations in agile contexts in which the project domains and the technological environments are new to the team, and the teams -with little experience in Agile contexts- have recently been created, so the team velocity is unknown.

In this scenario, we tried to answer the following research question: *when may the accuracy of an expert estimation made in a context of agile software development be improved by using historical data?* The results we obtained through our empirical study have led us to conclude that historical data may improve the accuracy of an intuitive estimation made by an expert when the estimator has limited experience in the job to be performed, the technologies to be used and the domain to be dealt with, and when the team velocity is unknown.

In section two, we will introduce three estimation methods: Expert Estimation (ExE), Analogy-Based Method (AbM), and Historical Productivity (HP). In section three, we will describe an empirical study and analyze the results obtained. In section four, we will investigate related work to see if there is any other evidence of improvement in expert estimation accuracy when using historical data, and finally, in section five, we will draw conclusions as regards the evidence of the benefits of using historical data.

II. Estimation Methods

This section will describe the three estimation methods used in our empirical study: ExE, AbM and HP. However, before doing so, it is important to focus on the definition of certain expressions used to define such methods. For example, when defining expert, Jorgensen [1] used a broad definition of the phrase, as he included estimation strategies that ranged from unaided intuition ("gut feeling") to expert judgment supported by historical data, process guidelines, and checklists ("structured estimation"). In his view, for an estimation strategy to be included under the expert estimation category, it had to meet the following conditions: first, the estimation work must be conducted by a person recognized as an expert in the task, and second, a significant part of the estimation process must be based on a non-explicit and non-recoverable reasoning process, i.e., "intuition". In our study, however, a narrower definition of the concept of expert was used: that which refers only to intuition. This way, we made a difference between intuitive ExE, and the methods that involve the use of historical data: AbM and HP. It is important to note that in our study, when we used Planning Poker –an ExE method-, no historical data was taken into account.

To further clarify the terms used, we must say that by AbM we meant the estimation performed by an expert, who is aided by a database containing information about finished projects [4]. As regards HP, which is another way of using historical data, it is worth mentioning that in our empirical study we focused on the size characteristic of the products, as suggested by one of the authors that inspired this article [4].

A. Expert Estimation Method (ExE)

When estimating the effort of a software development task, an expert estimation may be obtained either by a single expert, whose intuitive prediction will be considered an expert judgment, or by a group of experts, whose estimation will combine several experts' judgments.

A very frequently used way to obtain group expert judgment is called Planning Poker, a technique that combines expert opinion, analogy, and disaggregation. It is based on the consensus that is reached by the group of experts who are performing an estimation; in fact, it is considered a manageable approach that produces fast and reliable estimations [3][5][6]. This method was first described by James Greening [8] and it was then popularized by Mike Cohn through his book "Agile Estimating and Planning" [3]. It is mainly used in agile software development, especially in Extreme Programming [7]. To apply Planning Poker, the estimation team should be made up of, ideally, all the developers within the team, that is, programmers, testers, analysts, designers, DBAs, etc. It is important to bear in mind that, as this will happen in Agile contexts, the teams will not exceed ten people [3]. In fact, Planning Poker becomes especially useful when estimations are taking too long and part of the team is not willing to get involved in the estimation process [8]. The basic steps of this technique, according to how Grenning described them, are:

"The client reads a story and there is a discussion in which the story is presented as necessary. Then, each programmer writes his estimation on a card, without discussing his estimation with anyone else. Once every programmer has written down his estimation, all the cards are flipped over. If everybody has estimated the same, there is no need for discussion; the estimate is registered and the next story is dealt with. If the estimates are different, the team members will discuss their estimates and try to come to an agreement" [8].

Mike Cohn further developed this technique: he added a pack of cards especially designed to apply this technique and he shaped the whole process: each estimator is given a pack in which there are cards that have numbers written on them Those numbers represent a valid estimation, such as 0, 1, 2, 3, 5, 8, 13, 20, 40, and 100. Each pack has to be prepared before the Planning Poker meeting and the numbers should be big enough to be seen from the other side of the table. There is a raison d'être for the estimation scale presented above. There are studies which have demonstrated that we are better at estimating things which fall within one order of magnitude [9][10], so these were the cards that were employed when Planning Poker was used in the empirical study reported in this article. It should be noted that no historical data was used when estimating with Planning Poker for our study.

B. Analogy-Based Method (AbM)

The idea of using analogy as a basis to estimate effort in software projects is not new: in fact, Boehm [11] suggested the informal use of analogies as a possible technique thirty years ago. In 1988, Cowderoy and Jenkins [12] also worked with analogies, but they did not find a formal mechanism to select the analogies. According to Shepperd and Schofield [13], the principle is based on the depicting of projects in terms of their characteristics, such as the number of interfaces, the development methodology, or the size of the functional requirements. There is a base of finished projects which is used to search for those that best resemble the project to be estimated.

So, when estimating by analogy, there are *p* projects or cases, each of which has to be characterized in terms of a set of *n* characteristics. There is a historical database of projects that have already been finished. The new Project, the one to be estimated, is called "target". Such target is characterized in terms of the previously mentioned *n* dimensions. This means that the set of characteristics will be restricted to include only those whose values will be known at the time of performing the prediction. The next step consists of measuring similarities between the "target" and the other cases in the *n*-dimensional space [14].

Such similarities may be defined in different ways, but most of the researchers define the measuring of similarities the way Shepperd & Schofield [13] and Kadoda, Cartwright, Chen & Shepperd [14] do: it is the Euclidean distance in an *n*-dimensional space, where *n* is the number of characteristics of the project. Each dimension is standardized so that all the dimensions may have the same weight. The known effort values of the case closest to the new project are then used as the basis for the prediction.

In our empirical study, we applied AbM in its simplest version. The participants compared the user stories of two projects: one considered "historical" and the other one "target". The Estimated Effort (EE) of the user story of the target project was, in fact, the Actual Effort (AE) of the "most similar" user story of the historical project. Actually, no specific characteristics of the user stories were specially taken into account.

C. *Historical Productivity*

Jørgensen, Indahl, and Sjøberg [4] defined Productivity as the quotient of Actual Effort (AE) and Size, and the EE as the product of Size and Productivity. In this empirical study, COSMIC [15] was used as a measure of Size, and EE was calculated as the product of Size and Historical Productivity (HP). The HP is the value of productivity of the project to be used as historical project, that is, the quotient of the AE and the Size of the historical project.

To measure size, COSMIC was selected because it is an international standard [16] that is widely recognized in the software industry, and also because there is a previous study that used it in an Agile context [17]. With the COSMIC software method, the Functional User Requirements can be mapped into unique functional processes, initiated by functional users; in fact, user stories are actually used in this paper. Each functional process consists of sub-processes that involve data movements. A data movement concerns a single data group, i.e., a unique set of data attributes that describe a single object of interest. There are four types of data movements: a. an Entry moves a data group into the software from a functional user, b. an Exit moves a data group out of the software to a functional user, c. a Read moves a data group from persistent storage to the software, and d. a Write moves a data group from the software to persistent storage.

In the COSMIC approach, the term "persistent storage" denotes data (including variables stored in central memory) whose value is preserved between two activations of a functional process.

The size expressed in CFP is given by the equation CFP = Entries + Exits + Reads + Writes, where each term in the formula denotes the number of corresponding data movements. So, there is no concept of "weighting" a data movement in COSMIC, or, equivalently, all data movements have the same unit weight.

III. Description of our Empirical Study

Our empirical study is described in this section, considering its conception, how it was planned, the particularities of its execution and the results obtained.

A. *Definition*

This empirical study was designed in order to establish when the accuracy of an expert estimation made in a context of agile development, under the circumstances that will be described below, may be improved by using historical data. Such circumstances are: the project domain and the technological environment must be new to the estimator, and the team would have recently been created, so that the team velocity will be unknown.

The development steps of this empirical study may be summarized as follows:

The study was developed in the context of graduate education for IT practitioners from different educational and work backgrounds. The participants attended a workshop which had two objectives, one oriented to the subjects and another one oriented to the development of this empirical study. The workshop gave the participants the opportunity to: a. understand both how a historical database is built, and under which circumstances such database will give value to the estimation process, b. estimate using three methods and c. compare their results with other participants' results. Later on, the same workshop was conducted for undergraduate students.

The workshop participants were asked to re-estimate the first spring of an application that had been previously developed by a group of undergraduate students who did not participate of the workshop. The selected application had been developed using a development language unknown by the workshop participants and the application belonged to a domain the latter knew little of. The original team velocity was not reported to the participants, to simulate that it was unknown.

The re-estimations were made using three different estimation methods: ExE, based on the participants' intuition, and two other methods which use historical data. The historical data was obtained from a similar application that had been developed by a third undergraduate group –a group that had neither developed the original application nor participated of our empirical study-.

To guarantee the best results, we followed the recommendations of Juristo and Moreno [18] and Wohlin et al. [19] in order to develop this empirical study. To report it, we took into account Jedlitschka, Ciolkowoski and Pfahl's guidelines for reporting empirical research in software engineering [20].

As previously stated, the objective of this empirical study was to analyze when the accuracy of an estimation made by an expert, based on his personal intuition, may be improved by using historical data. This objective was achieved by comparing the estimation errors obtained by two different groups: undergraduate students and practitioners, when estimating using three different methods: ExE, AbM, and HP.

In fact, the hypotheses to be tested were:

H_0: The mean value of the MRE calculated with the ExE is equal to the mean value of the MRE obtained when calculating with AbM or HP.

H_1: The estimated mean value of the MRE calculated with the ExE is lower than the mean value of the MRE obtained when calculating with AbM or HP.

B. Planning

The *experimental subjects* were IT graduate students and undergraduate advanced students of Informatics Engineering. In fact, all of the graduate students were practitioners. So, in this paper, when we say "participants" we mean both the graduate and undergraduate students, and by "practitioners" we refer only to the graduate students.

The participants were asked to give some information about themselves regarding the following aspects:

- If graduate or undergraduate student
- Professional experience (they had to state the number of years they had worked in software development)
- Experience with COSMIC
- Experience with user stories (they had to inform the number of user stories that they had written/read (fewer than 20, 20-100, more than 100)
- Experience with Ruby [21] language.
- Experience in Database development
- Experience in working in Agile development contexts.
- Level of prior knowledge about the productivity of the teams that developed the experimental objects (high, medium, low)
- Level of experience in the technologies used to develop the experimental objects (high, medium, low)
- Level of experience in the domain of the experimental objects (high, medium, low)

The *experimental objects were* two similar applications (P1 and P2), which were social networks. The first application was a system through which users may conduct surveys. The system classifies users into several categories, builds different groups and instantly surveys those users who fall within the right categories. It was developed by a team of undergraduate students who registered the estimated and actual hours using the Scrumy tool [22], and who were supervised by two professors.

The second application, which we identified as the "target project", was a network where different types of events may be published. For example, an event may be a party, a meeting or a football game. Events are the core elements in this application, not people. It works with event and friend suggestion algorithms and gives the option of buying a ticket for an event online.

The data corresponding to the experimental objects are displayed below. Table 1 shows the user stories of P1 and the Actual Effort (AE) of each user story measured in man hours. As some user stories were not functional processes, they were discarded. Table 2 shows the user stories of P2, which are the user stories of only the first sprint, as it was the only sprint for which effort was estimated.

As regards the counting of the man-hours worked on P1 and P2, one of the tasks within the assignment the undergraduate students that developed the projects had to undertake was to register the hours worked. These two groups did not participate in the empirical study; in fact, they were undergraduate students from a university different from the one where the undergraduate participants studied. The applications were developed in an Agile context, as an assignment in a practical subject. They first estimated the work to be done and then compared their estimations to their real effort. Two professors supervised these tasks. This empirical study used the actual effort of P1 and P2 and the estimated effort of P2 (obtained by the original development group), so that they may be compared to the participants' results.

The aspects of the development process that were controlled to facilitate such comparison were:

- Similarity: Two similar applications that had been developed in Agile contexts were selected as experimental objects. They had been developed in an academic context by advanced undergraduate students, who had been requested to develop an application for an assignment in which a company environment was simulated.
- Experience in team velocity: Since in Agile contexts developers learn from previous estimations, and in this case the estimators were expected to have no previous experience, only the first sprint of the target application could be estimated in order to be compared to the actual effort estimation of P2, as it was only for the first sprint that the original P2 estimators did not have experience in team velocity.
- Language experience: Participants with experience in Ruby language, in Agile contexts, and / or COSMIC were equally distributed.

In order to obtain comparable results in this study, man-hours had to be used to unify the unit of measurement of effort, as the historical values had been previously measured in man-hours, instead of in story points or ideal hours, which are the measures usually used to make effort estimations with Planning Poker in Agile contexts [3].

The workshop was run following these steps:

1) The participants were given a set of materials that included: Brief Vision Documents [23] of P1 and P2, the professor's slides explaining the empirical study, and an Excel file where each sheet was a step of the empirical study.

TABLE I. DATA OF THE APPLICATION TO BE USED AS HISTORICAL INFORMATION

P1	Actual Effort [man-hours]
Create survey	18
Sign up	15
See user's profile	9
Answer survey	9
Log in/Log out	6
Comment on survey	12
Search for survey	9
Eliminate user	3
Edit personal data	6
Search for user	9
Generate and publish statistics	30
Follow user	30
Select user segment	18
Sort the content according to date	18
Upload pictures	21
UPR (User Popularity Ranking)	36

TABLE II. USER STORIES OF THE TARGET APPLICATION

P2
Create, Modify and Eliminate User
Log in (Log out)
Create event
Search for event

2) Each one of the empirical study steps was explained to the participants. The participants were trained to perform each activity. Also, two examples of COSMIC measurement were included.

It is important to note that the participants worked with an Excel file that was designed to facilitate the understanding of the activities, and the sequence in which they had to do them. The following are the activities presented sequentially in each one of the sheets in the file:

a) *Perform the expert estimation,* based on their intuition, they estimated the man hours to be worked on the target application (P2). Based on the Vision Document of P2, the participants estimated the EE of each user story described in Table 2.

b) Build the historical database. The participants measured the size of the user stories of the historical application (P1) by using COSMIC, as shown in Table 1. The Excel sheet automatically calculated the Historical Productivity (HP) of P1 as the quotient of AE_{P1} and $Size_{P1}$, where AE_{P1} is equal to the sum of the AE of each user story of P1, and $Size_{P1}$ is equal to the sum of the Size of each user story of P1. The data movements of P1 were indentified for each user story, based on: the information included in the Vision Report, the name of the user story, and the explanation given by the leader of the workshop when asked for it. The measurement of the user stories, using COSMIC, was performed in a way similar to that of [17].

c) *Measure the size of the target application (P2), by using COSMIC to measure the size of the user stories.* These size values were automatically used to calculate $EE_{P1,}$ which was calculated as the product of $Size_{P1}$ and Historical Productivity (HP_{P2}).

d) *Estimate the effort for the target application (P2) using AbM.* The participants had to select for each one of the user stories in P2 the most similar user story from the set of user stories in P1 -though based on their characteristics, not on their size- and then assign to the Estimated Effort (EE) of each user story in P2 the AE of the similar user story in P1.

e) *Individually compare and analyze the EE values obtained using ExE, AbM, and HP methods*. The Excel sheet automatically presents a Table which displays the three EE values –those obtained by applying the three different estimation methods- for each user story in P2.

3) The participants estimated the effort of the target application following the steps listed above, and completed the worksheets.

4) The data was collected and the results were analyzed with the participants. A rich discussion about the comparison of the MRE obtained by applying the three estimation methods (ExE, HP and AbM) was conducted by the leader of the empirical study.

C. Execution

The characteristics of the participants are described in Table 3.

Forty nine undergraduate students, who were distributed in fourteen groups of 3-4 students, participated in the two workshops. The median work experience of the students was three years. No one had experience using COSMIC, and they had little experience with user stories. All of them had approved the course "Database" and only 8 had experience in working in an Agile context, that is to say, a small proportion of them. The Level of experience of the development teams in the technologies to be used and in the domain of the experimental objects was low. In one of the workshops, fourteen practitioners worked on their own. The median work experience of the practitioners was fourteen years. No one had experience in using COSMIC, and five of them had experience with user stories.

Their median experience in "Database" was ten years and only three of them had experience in working in an Agile context, which is a small proportion. The Level of experience in the technologies and in the domain of the experimental objects was medium-low.

Table 4 shows the effort estimation values of the target project, obtained by the two groups applying the three estimations methods: ExE, HP, and AbM. Moreover, the AE of the student group that developed the target application (P2) was 35 man-hours.

Figure 1 shows the boxplots of the residuals and Figure 2 the boxplots of the MRE for the target project. To obtain the MRE, the actual value registered for the first sprint of P2 by the group that actually developed the project was used as AE.

TABLE III. WORKSHOP PARTICIPANTS

Type	Number	Work Expe-rience (Years)	Expe-rience using COSMIC	Number of User Stories [<20, 20<US<100, >100]	Database Experience	Experience with Ruby Language	Work expe-rience in Agile context	Experience in the technologies	Expe-rience in the domain
Under graduate	49 (14 groups)	[0-13] Median: 3	No one	<20: 44 20<US<100: 3 >100: 2	All had approved the Course "Database"	No one	Only 8	Low: 47 Average: 2 High: 0	Low: 43 Average: 4 High: 2
Practi-tioners	14	[4-36] Median: 14	No one	<20: 9 20<US<100: 3 >100: 2	Database experience measured in years [0-36] Median:10	Only one	Only 3	Low: 9 Average: 5 High: 0	Low: 11 Average: 3 High: 0

The boxsplots show the different results obtained by each group of participants. The undergraduate participants obtained better estimation results when applying the AbM, rather than the ExE and HP methods. Figure 2 shows the median values, but it must be noted that a more significant difference was observed when comparing the values obtained for the mean MRE in the undergraduate group: AbM: 69.80, ExE:151,43 and HP:175,04. On the other hand, the practitioners group obtained the best results when applying ExE, instead of HP and AbM, as shown by the boxplots. Also, their mean values were ExE: 29.09, HP: 205.16, and AbM: 87.14.

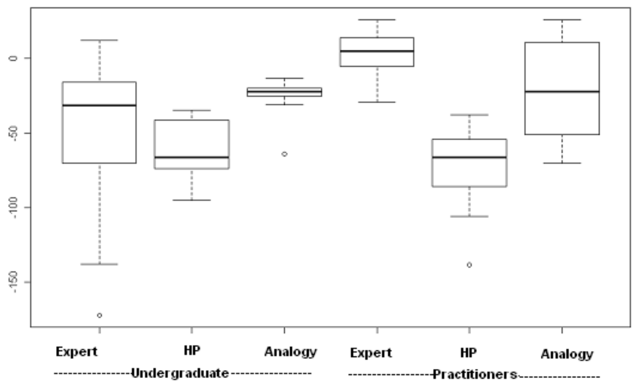


Fig. 1. Boxplots of the residuals of the target project

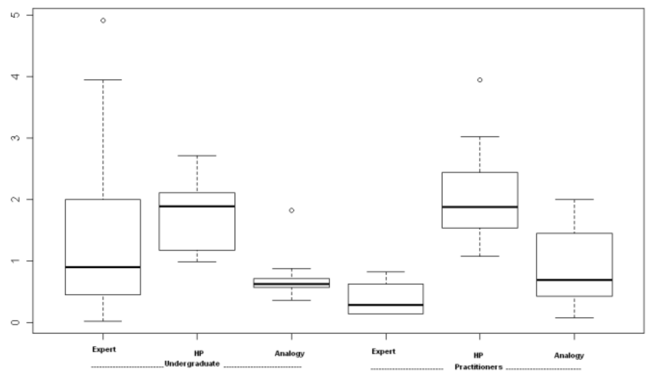


Fig. 2 Boxplots of the MRE of the target project

D. Threats to validity

The difference in background of the experimental subjects is the major weakness of this empirical study. However, this drawback may be transformed into a strength if we consider that in this empirical study the experience of the expert is stressed, showing that the accuracy of an expert estimation depends on the estimator's expertise, which is measured by his work experience, his level of experience in the technologies used to develop the experimental objects and his level of experience in the domain of the experimental objects.

Another threat is that the expert estimations were made in two different manners: either alone or in groups. The practitioners worked alone and the undergraduate students formed groups of three or four persons and used Planning Poker to obtain the expert values. However, we think that this combination of expert methods, that is, using Planning Poker or not, did not introduce bias in this study, in accordance with what was reported in [24].

Unfortunately, only a brief explanation about COSMIC was given to the undergraduate students, since it was not possible to give an extensive explanation, as there was not enough time to do so (the whole workshop was three hours long). Thus, the little available time was devoted to those COSMIC characteristics that were necessary for them to know in order to make a correct measurement. However, this did not seem to be a big problem, as the concept of data movement is quite intuitive for all the participants and the medians of the errors shown in both Figure 1 and 2 for the HP method are similar.

Also, the use of examples and previous training in Function Points made it easier for the participants to understand how to use this measuring method. On the other hand, the practitioners had been previously trained in COSMIC, so they presented no difficulty. Besides, if anybody had any doubts, the person who led the empirical study would give further explanations.

The order in which the estimations were performed could have introduced bias in the result, so it would have been more convenient if the participants had not performed the estimations in the same order, except for ExE, which must always be performed in the first place.

The selection of a similar application to make up the historical database is clearly an advantage in order to obtain a better estimation, but the problem is that sometimes the estimator does not have data about similar applications at hand, so she or he has to use an application from a different domain. This circumstance may vary the results obtained in this empirical study.

The experimental subjects were identified either as undergraduates or practitioners. However, it may be argued that more categories would have been necessary, as some of the practitioners had more experience in the domain or in the technologies than some others. Consequently, to obtain more evidence of the benefit of using historical data, it is necessary to have a bigger number of estimators, which would allow us to identify different levels of expertise, for example, three expertise levels for practitioners and three for undergraduates.

To conclude, as the experimental objects used in the empirical study came from a particular environment and the experts' experience did not cover the big spectrum of expertise that exists, general conclusions cannot be drawn because there may be different estimation problems in different environments and experts' performances.

IV. DATA ANALYSIS AND INTERPRETATION

To answer the research question posed above, it is important to understand the circumstances under which the use of historical data may improve the expert estimation accuracy. In this empirical study, two types of experts were involved: we called them undergraduate and practitioner participants. Consequently, each group will be separately analyzed first, then the statistical significance of the results will be dealt with, and afterwards, the research question will be answered. Finally, this will be completed with the discussion of aspects omitted in the previous sections.

A. *Result analysis*

1) *Undergraduate*

We noticed that there were three aspects that affected the intuitive expert estimation: the work experience, the level of experience in the technologies used to develop the experimental objects, and the level of experience in the domain of the experimental objects. The undergraduate participants' work experience measured in years varied from 0 to 13, with a median of 3. This shows that the "experts" had little experience in estimations and also, that the level of experience in the technologies used and in the domain was low.

Their best result was obtained when using AbM: the MRE median was 63% within a [37%-183%] range. The lack of experience, in this case, was compensated for by the historical data.

By using HP, the MRE dispersion was increased: the MRE values ranged from [99%-272%]. The MRE of the 14 groups had a median of 189% and a standard deviation of 54%.

2) *Practitioners*

When compared to the undergraduate participants, the most significant difference was their work experience: measured in years, it varied from 4 to 36, with a median of 14. Ten practitioners were project leaders or managers, three were senior developers and only one was a junior developer. This shows that these "experts" had experience in project management and, of course, in estimations.

TABLE IV. EE OF THE TARGET PROJECT

Participants	**Number of estimations**	**ExE %**	**HP %**	**AbM %**
Undergraduates	14 (made by groups of 3-4 undergraduate students)	161.00	110.00	57.00
		61.00	74.70	57.00
		34.00	76.30	60.00
		65.00	69.72	48.00
		207.00	84.12	48.00
		85.00	106.13	55.00
		173.00	90.21	66.00
		68.00	102.84	57.00
		79.00	101.15	57.00
		56.00	101.44	51.00
		51.00	72.00	57.00
		32.00	130.15	57.00
		105.00	108.93	99.00
Practitioners	14	11.00	108.94	11.00
		30.00	173.22	24.00
		21.00	84.77	20.00
		30.00	122.15	60.00
		9.00	90.55	9.00
		64.00	85.96	39.00
		30.00	120.61	105.00
		29.00	111.87	86.00
		16.00	72.88	32.00
		30.00	105.05	95.00
		40.00	88.93	57.00
		40.00	97.07	94.00
		49.00	92.37	70.00
		57.00	140.94	57.00

The practitioners' level of experience in the technologies used to develop the experimental objects and the level of experience in the domain of the experimental objects was medium-low. These characteristics justify the results obtained when using ExE.

During the study, three of them did not perform the expert estimation because they considered that they were no "experts", while two of them assigned to the expert estimation the same value they had assigned to the AbM estimation. Seven of the eleven practitioners that applied pure expert estimation estimated with an MRE lower than 25%.

The estimation by AbM had a MRE median of 70 % in a range result of [8.57%-200%], which is a result similar to that obtained by the undergraduates.

By using HP, the MRE dispersion was increased: [108.22%-384.91%]. The MRE of the 14 practitioners had a median of 189% -similar to that of the undergraduate value-and a big standard deviation of 75%, which may have been caused by the subjectivity introduced by COSMIC, originated by the practitioners' different backgrounds.

3) *The statistical significance of the results*

The Wilcoxon rank test, at a significance level of 0.05, was used to analyze the statistical significance of the results. This non-parametric test was selected because the distributions of the variables were not normal. It was applied to test the accuracy of ExE versus that of HP or AbM, according to the results obtained by each group (practitioners and undergraduate participants). The MRE and the absolute residuals were used. Table 5 shows the p-value of each subset, when using the MRE. The results obtained when using the absolute residuals are not shown because there is no significant difference.

TABLE V. STATISTICAL SIGNIFICANCE

Groups	ExE vs:	MRE
Undergraduate	HP	0.162
	AbM	0.948
Practitioners	HP	0.000
	AbM	0.022

When analyzing the MRE obtained by:

- the practitioners, when comparing ExE to HP, it was possible to reject H0 in favor of H1.
- the practitioners, when comparing ExE to AbM, once again, it was possible to reject H0 in favor of H1.
- the undergraduates, when comparing the ExE method to HP, it was not possible to reject H0 in favor of H1.
- the undergraduates, when comparing the ExE method to AbM, it was not possible to reject H0 in favor of H1.

It should be noticed that the three practitioners who did not use the method, as they did not consider themselves to be “experts”, were also included in the table. However, later on, the Wilcoxon rank test was also computed, but this time only for the eleven practitioners who made the estimations, and the results did not vary.

Now we can answer the research question: W*hen may the accuracy of an expert estimation made in a context of Agile software development be improved by using historical data?*

These results show that the expert estimation was not improved by the use of historical data when the expert had some work experience, and his level of experience in the technologies used to develop the application, and his level of experience in its domain were medium-low.

However, we found out that historical data may improve the expert estimation when the estimator’s work experience, his level of experience in the technologies used to develop the application, and his level of experience in the domain of the application to be developed is low.

4) Discussion

There are some aspects that have not been mentioned yet, but it is worth doing so now. One of them is the little experience in Agile development contexts that the two groups had. We think that this fact did not affect the results obtained because, as the work experience of the undergraduate group was small, their experience in Agile contexts was small too. On the other hand, practitioners were experienced in project management and estimations, so this compensated for their little experience in Agile contexts. On top, as the empirical study was designed to only use the first sprint of a software product development, no estimations were made for the rest of the sprints -which would be usually done when using an Agile method- so their little experience in Agile contexts had no impact on our study.

Another interesting aspect is that most of the effort calculations proved to be underestimated, which may be seen in Figure 1. This could be explained by the fact that almost all the participants did not have previous experience with the Ruby language. On the contrary, the group that developed the target application had previous knowledge of the velocity that they could achieve because they had done a Ruby on Rails tutorial before. Consequently, the level of experience of this group in the technologies used to develop the target application and the level of experience in the target application domain was medium-high, which justifies the accuracy of the estimation: 3% MRE, which was high. At the same time the group that developed the target application had a higher velocity than the group that developed the historical application. Obviously, the bigger the difference in the velocity, the bigger the error in the effort estimation.

One question that may arise is: how would the participants be able to make meaningfully expert estimations if they did not have any knowledge about the developers? This condition was part of the scenario that we were simulating; as it was stated in the introduction of this paper, the team velocity was unknown.

Figure 2 shows that the medians obtained by the two groups when estimating with HP were similar, but their standard deviations were not: the standard deviation of the MRE for the undergraduate group was 53.7 and 75 for the practitioners. This is a consequence of the subjectivity introduced by the COSMIC measurement of both the historical user stories and the user stories to be estimated. The estimation was affected by the subjectivity of the measurements and by the difference between the historical productivity of P1 and the actual productivity of P2.

Figure 2 shows that the MRE medians obtained when the two groups used the AbM method were similar but their MRE distributions were quite different. It was surprising to see that the results obtained by the practitioners using the AbM were worse than those obtained by the undergraduates. As the AbM is based on the selection of a “similar” user story, we may conclude that the undergraduate participants had a comparable concept of “similarity” to that of the original undergraduate group that developed the target application.

The estimation results obtained with the AbM and HP method would have been better if the historical data had been obtained from a similar project –one developed using Ruby on Rails- , but unfortunately, there was none available. Besides, the fact that the user stories that were not functional processes were discarded may have also influenced the results. In addition, another interesting factor that may have been considered is team size.

In our study, the empirical objects were two similar applications, but what would have happened if they had not been similar? Obviously, the results of the undergraduate group would have been affected, as their best results were obtained using the AbM. The reason is that such method is based on analogy, so if the degree of similarity between the application from where the historical data was to be obtained and that of the target application had been low, the accuracy of the estimation would have been poor too.

Moreover, although we only used the estimates of the first sprint of the target application this time, we believe the estimates of the following sprints could be used in future replications to evaluate if (and to what extent) expert estimations improve while participants gain knowledge of the projects (while AbM and HP are expected to yield constant accuracy throughout the sprints).

Finally, we may wonder about the participants' characteristics included in Table 3 and the reason why other characteristics were not included. To begin with, database experience is related to work experience, so it was necessary to check it because the COSMIC measurement would have been affected if experience in database had been small. In fact, the experience in using COSMIC was defined as a controlled variable. Moreover, the number of user stories the participants had written/read was included because it is related to their work experience in Agile contexts: in fact, there was a correlation between the number of user stories read/written and their experience in Agile contexts, which proved the consistency of the information. In addition, the level of experience with Rugby language and the level of experience in the technologies to be used had to be tested in order to verify if the participants fit our empirical study. Besides, the impact of the level of experience in the application domain was previously analyzed by [25]. We think that these characteristics have made the main differences between the two groups clear.

V. RELATED WORK

Apparently, this has been the first article to have been written about whether using historical data in an agile context improves expert estimation.

However, regarding expert estimation in general, there are some authors that have already reported evidence about the importance of the developers' level of maturity when evaluating the accuracy of estimations, which is in line with the conclusions of our study. For example, SCRUM pioneers believe it is acceptable to have an average error rate of 20% in their results when using the Planning Poker estimation technique, but they have admitted that this percentage depends on the level of maturity of the developers [25]. Another study [26] agrees with this statement, as it indicates that the optimism bias which is caused by the group discussion diminishes or even disappears as the expertise of the people involved in the group estimation process increases.

On the other hand, another study [27] has already examined the impact of the lack of experience of the estimators in the domain problem, as well as that in the technologies used in a software development project. In fact, what was studied was the accuracy with which the effort of a given task was estimated. Such estimation was performed by a single expert by comparing the estimated and the actual efforts. The reason for researching on this aspect is that, occasionally, organizations do not have in their staff experts that have relevant prior experience in some business or technology related aspect of the project they are working on. This research investigates the impact of such incomplete expertise on the reliability of estimates.

It is important to note that Jorgensen [1] has both defined a list of twelve "best practices", that is to say, empirically validated expert estimation principles, and suggested how to implement these guidelines in organizations. One of the best practices he proposed is to use documented data from previous development tasks and another one is to employ estimation experts with a relevant domain background and good estimation records. Actually, our article headed in the same direction; we focused on historical data and we analyzed the impact of the difference in experts' skills.

An aspect that should be taken into account when performing expert estimations is excessive optimism, as it is one of the negative effects that influences the most when a software project fails. Jørgensen and Halkjelsvik [28] have discovered something that seems to be important to understand what may be leading estimators to excessive optimism: the format used to word the question that asks about effort estimation. The usual way to ask about effort estimation would be: "How many hours will be used to complete task X?". However, there are people who would say: "How many tasks could be completed in Y hours?". Theoretically, the same results should be obtained by using any of the two formats. Nevertheless, according to Jørgensen and Gruschke [29], when the second option is used, the estimations which are thus obtained are much lower than those obtained when the traditional format is used, that is to say, the time to fulfill a task will be shorter, and consequently, the estimation will be much more optimistic. Thus, in our study, the expert estimations were made using the usual question. In fact, the final recommendation of this study is that the traditional format should always be used, as this does not contain any deviation imposed by the clients who ask the developers for more than they can pay for.

VI. CONCLUSION AND FUTURE WORK

This paper specifically focuses on an agile context in which the project domain and the technological environments are new to the estimators, the teams have recently been created, and the team velocity is unknown. As under these circumstances historical data may become important, we tried to answer the following research question: *when may the accuracy of an expert estimation made in a context of agile software development be improved by using historical data?* To find out whether there is any advantage in using historical data when the historical velocity is unknown, an empirical study was developed in an Agile software development context.

Historical data seems to be valuable when the work experience, the level of experience in the technologies to be used to develop an application, and the level of experience in the domain of the application to be developed are low.

So, for estimators who have the restrictions described above, and who have no option but to work with them, we may suggest the following:

- Use intuitive expert estimations when your work experience, your level of experience in the technologies to be used to develop the application, and your level of experience in the domain of the application to be developed are not low.
- Use historical data when your work experience, your level of experience in the technologies to be used to develop the application, and your level of experience in the domain of the application to be developed are low.

In order to generalize this conclusion, a replication of this empirical study is recommended, especially if different software life cycle models [30], application domains, expert profiles, and levels of performance are included. Also, different estimation methods, such us linear regression or Analogies –next time, using the size characteristic- may be used. Finally, in order to enrich this empirical study, it would also be convenient to compare the estimation performed by an expert who has deep knowledge of this domain, and also knows the team velocity, to the estimations obtained by the participants of our study.

AKNOWLEDGMENTS

Our thanks to the Research Fund of Austral University, which made this study possible, and to Luigi Lavazza for his opportune comments.

REFERENCES

[1] M. Jorgensen, “A review of studies on Expert estimation of software development effort,” Journal on System and Software, Vol. 70, No. 1-2, 2004, pp. 37-60.

[2] M. Jorgensen, and Shepperd, “A systematic review of software development cost estimation studies,” IEEE Transactions on Software Engineering, Vol. 33, No. 1, January 2007, 2007, pp. 3-53.

[3] M. Cohn, Agile Estimating and Planning. Addison-Wesley, 2005.

[4] M. Jørgensen, U. Indahl, and D. Sjøberg, “Software effort estimation by analogy and regression toward the mean,” Journal of Systems and Software, 68(3), 2003, pp. 253-262.

[5] T.J.Bang, “An Agile approach to requirement specification,” Agile Processes in Software Engineering and Extreme Programming, SE:35, VL:4536, Lecture Notes in Computer Science, G. Concas,E. Damiani, M. Scotto, G.Succi, Eds., Springer Berlin Heidelberg, 2007, pp. 193-197.

[6] J. Choudhari and U. Suman, “Phase wise effort estimation for software maintenance: an extended SMEEM model,” in Proceedings of the CUBE International Information Technology Conference, ACM, 2012, pp. 397-402,

[7] N.C. Haugen, “An empirical study of using Planning Poker for user Story estimation,” Proceedings of AGILE 2006 Conference, Computer Society, IEEE, 2006, 9 pp. – 34.

[8] J. Grenning,. “Planning Poker or how to avoid analysis paralysis while release planning,” 2002, DOI=, http://sewiki.iai.uni-bonn.de/_media/teaching/labs/xp/2005a/doc.planningpoker-v1.pdf: August, 2013.

[9] E. Miranda. “Improving Subjective estimates using paired comparisons,” IEEE Software, 18(1), 2001, pp. 87–91.

[10] T. Saaty, Multicriteria decision making: the Analytic Hierarchy Process. RWS Publications, 1996.

[11] B. Boehm. Software Engineering Economics. Prentice Hall. 1981.

[12] A.J.C. Cowderoy and J.O. Jenkins, “Cost estimation by analogy as a good management practice,” in Proc. Software Engineering 88. Liverpool: IEE/BCS, 1988, pp. 80-84,

[13] M. Shepperd, and C. Schofield, “Estimating Software Project Effort Using Analogies,” IEEE Trans. on Software Eng., vol. 23, no. 11, 1997, pp. 736-743.

[14] G. Kadoda, M. Cartwright, L. Chen, and M. Shepperd. Experiences using Case-Based Reasoning to predict software project effort. Empirical Software Engineering Research Group Department of Computing Bournemouth University Talbot Campus Poole, BH12 5BB, UK. 2000.

[15] COSMIC – Common Software Measurement International Consortium, 2009, The COSMIC Functional Size Measurement Method - version 3.0.1. Measurement Manual (The COSMIC Implementation Guide for ISO/IEC 19761: 2003).

[16] ISO (2011) ISO/IEC19761:2011, Software Engineering -- COSMICFFP– A Functional Size Measurement Method, ISO and IEC.

[17] J. Desharnais, L. Buglione, and B. Kocatürk,. “Using the COSMIC method to estimate Agile user stories,” *in* Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement, ACM, New York, 2011, pp. 68-73.

[18] N. Juristo and A.M. Moreno, Basics of Software Engineering Experimentation. Kluwer Academic Publishers., 2001.

[19] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, and A. Wesslen, Experimentation in Software Engineering: an Introduction. Kluwer Academic Publisher, 2000.

[20] A. Jedlitschka, M. Ciolkowoski and D. Pfahl, “Reporting experiments in Software Engineering,” in Guide to Advanced Empirical Software Engineering, Section II, 2008, pp. 201-228.

[21] Ruby on Rails. http://rubyonrails.org/: August, 2013.

[22] Scrumy, http://www.scrumy.com: August, 2013

[23] K. Bittener and I. Spence. Use case Modeling. Addison Wesley, 2003.

[24] K. Molokken-Ostvold, N.C. Haugen and Benestad, H.C., “Using planning poker for combining Expert estimates in software projects,” Journal of Systems and Software, 81 (12), 2008, pp. 2106-2117.

[25] O. Ktata, and G. Lévesque, “Designing and implementing a measurement program for Scrum teams: what do agile developers really need and want?,” in Proceedings of the Third C* Conference on Computer Science and Software Engineering (C3S2E '10), ACM, New York, NY, USA, 2010, pp. 101-107.

[26] V. Mahnič and T. Hovelja, “On using planning poker for estimating user stories,” J. Syst. Softw. 85, 9 (September), 2012, pp. 2086-2095.

[27] S. Halstead, R. Ortiz, M. Córdova, and M. Seguí, “The impact of lack in domain or technology experience on the accuracy of Expert effort estimates in software projects,” in Proceedings of the 13th international conference on Product-Focused Software Process Improvement (PROFES'12), Springer-Verlag, Berlin, Heidelberg, 2012, pp. 248-259.

[28] M. Jorgensen, and T. Halkjelsvik, “The effects of request formats on judgment-based effort estimation,” Journal of Systems and Software, 83 (1), 2010, pp. 29-36.

[29] M. Jorgensen and M. Gruschke, “The Impact of lessons-learned sessions on effort estimation and uncertainty assessments,” IEEE Transactions on Software Engineering, Jan. IEEE computer Society Digital Library. IEEE Computer Society, 2009, pp. 368 - 383.

[30] A. M Davis, E. H. Bersoff and E. R. Comer, “A strategy for comparing alternative software development life cycle models”, Software Engineering, IEEE Transactions on (Volume: 14 , Issue: 10), 1988, pp. 1453 – 1461.

Agile-User Experience Design: an Agile and User-Centered Process?

Lou Schwartz
Public Research Centre Henri Tudor
Luxembourg, Luxembourg
lou.schwartz@tudor.lu

***Abstract*—Agile-User Experience Design, also called Agile-UX, is a trend of the last decade that mixes values and practices from the Agile software engineering methods and the User-Centered Design. Several practitioners have proposed different processes to organize the work between development and design. After a short reminder of the values of Agile and User Centered Design methods, this paper presents five processes proposed in the literature. The processes are discussed with regards to their respect of the Agile and User Centered Design values. This comparative study concludes that not one process totally covers the Agile and User Centered Design values: they all make a trade-off and could be completed by practices and by a state of mind and a willingness adopted by the team.**

Keywords-Agile; Agile-UX; Agile Software Techniques; Software Engineering; User-Centered Desing;

I. INTRODUCTION

Since a decade, several software companies, or only at the teams' level, try to integrate Agile software development methods and User Centered Design (UCD) [6][8][14][19]. This integration, called Agile-User Experience Design or Agile-UX, is bound on the one hand to the interesting performance of Agile methods to quickly provide software that answers the users' needs with a certain level of quality, and on the other hand it results in the observation that this software quality is relative, particularly related to Human Computer Interactions aspects [3][18]. Based on this observation, several practitioners tried to integrate UCD in their Agile process with various degrees of success. After a reminder of Agile and UCD methods in section II and III, this paper will present processes used to integrate Agile and UCD, often addressed in the literature in section IV and discuss them regarding their respect of the agile and UCD values in section V.

II. AGILE METHODS

The Agile methods' goal is to enhance the value of the delivered product in order to satisfy the customer's requirements. Agile methods adopt the following four values defined in the Agile Manifesto [1]:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

The Agile movement was instigated and pioneered by software developers in reaction to a frustration emerging from history of delayed projects, budget overruns and stressful jobs [2]. For the Agile Manifesto founders, these problems have their origin in a too big analysis, specification and design done before code writing that enables volatile or useless requirements and incompleteness. With the Agile methods, customers would obtain faster working software that corresponds better to their real requirements, thanks to the flexibility provided to the development process [2].

Agile methods are focused on the developers' work and on the development quality [4]. Even if the aim of Agile methods is to satisfy the product owner's (who is the representative of stakeholders: customers and end-users) requirements, they define neither method nor good practices to achieve this objective, particularly for the needs elicitation or the design part. The needs elicitation is done by the product owner, based on his own knowledge of the domain or of the work done by users. He can use the methods he wants, including involving the users (e.g., by interviews, context inquiries, etc.). The user interface design depends on the openness to ergonomics of developers, customer and users. So there is no guarantee about it. [4]

The use of the UCD principles and methods is one way to ensure answering to users' needs. Based on these assessments, Agile teams can benefit from the integration of UCD methods with Agile to improve, in particular, the needs elicitation and the design part.

III. USER-CENTERED DESIGN

UCD focuses on producing usable software that satisfies real end-users and customers. This method, described by the standard ISO 9241-210 [9] defines the process to follow to produce software that meets the users' requirements. It includes in particular the design and the validation stages.

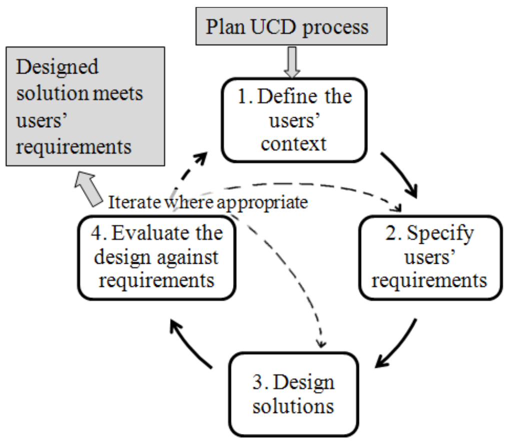

Plan UCD process
Designed solution meets users' requirements
1. Define the users' context
2. Specify users' requirements
3. Design solutions
4. Evaluate the design against requirements
Iterate where appropriate

Figure 1. UCD process as described by the standard ISO 9241-210 [9].

Four activities compose the UCD process (see Figure 1.):

- Understanding and specifying the context
- Specifying the user needs
- Produce design solutions to meet user requirements
- Evaluate the designs against requirements

The principles of the UCD are listed below [9]

- The design is based upon an explicit understanding of users, tasks and environments
- Users are involved throughout the design and the development
- The design is driven and refined by user-centered evaluation
- The process is iterative
- The design addresses the whole user experience
- The design team includes multidisciplinary skills and perspectives

Even if some Agile concerns could prevent a UCD attitude [4] (the focus is often more on programming techniques and programmers, automated tests, very short iterations and fast increments) a reconciliation of both approaches is possible and has often been implemented [6][8][11][12][14][15][19]. The integration of both methods implies focusing more on design activities. It results to a redefinition of the process to organize the activities dedicated to the design and the process dedicated to the development.

IV. Review of the Agile-UX processes presented in the literature

A major issue listed in the literature about Agile-UX is the organization of the work between development tasks and UCD tasks in respect to the Agile and UCD values. Among the existing work, five propositions of process design are studied in this section.

A. *Parallel tracks*

To manage exchanges and to organize the work to carry out between developers and usability experts, Sy [19] proposes that they work in parallel tracks after the planning iteration also called iteration “0”. It enables usability experts to keep ahead of the developers, to have enough time to gather users’ data, to analyze that data and to propose design solutions. For that, designers and developers work with one to two iterations of delay (see Figure 2.). During the iteration *i*, designers:

- Gather user and context data for the iteration *i*+2
- Work on the designs for the iteration *i*+1
- Help developers for the implementation of the designs of the iteration *i*
- Evaluate the software developed during the iteration *i*-1

The principle of parallel tracks is well acclaimed by usability experts who test it [6][15][19] thanks to the proactive attitude given to them. As any method, the Sy’s process has advantages and potential issues.

The advantages of working ahead of the development team [14] are:

- Better definition of the conditions of satisfaction (test acceptance criteria)
- Better planning the design
- Better inclusion of designs in the global users’ process
- Designers can be more concentrated on exceptions rather than trying to produce the best design right the first time.

The potential issues of parallel tracks are [14]:

- Sensation of not being one team that can give a vision of inequality
- Exclusion or self-exclusion of usability experts of some meetings
- Risks of the lack of communication which could lead to misunderstanding and resentment
- Forget to rectify issues noted during previous iteration’s tests.

To avoid these issues two solutions are proposed [14]: encourage communication, build common channels of communication; and give helpful assistance to developers as soon as possible when a design is not understood.

This iterative process covers the four UCD activities and it also respects the following UCD principles:

- Understanding of users, tasks and environment: the activities of gathering data on user and context are scheduled.
- Users’ involvement: users can be involved for the gathering of data and for design, but they are particularly consulted to test the developments.
- Evaluation: software tested by users.
- Iterative: intrinsic to the process.
- Multidisciplinary: by the involvement of designers, developers and stakeholders.

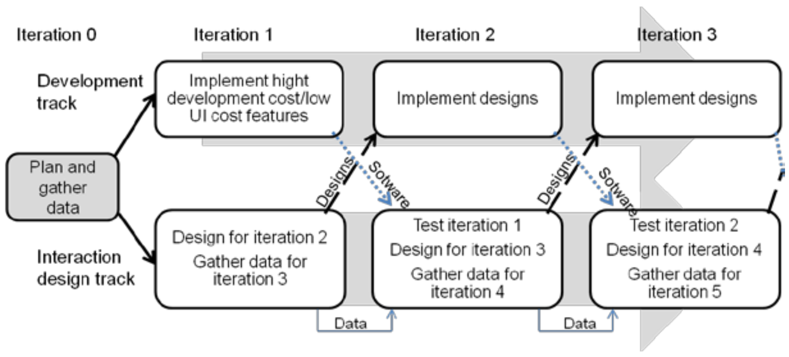


Figure 2. Sy’s parallel tracks of work [19].

B. *Design work done on parallel levels*

Armitage [2] proposes another type of parallel work that concerns only the designers' work organization. The design work is done on three parallel levels (see Figure 3.) from unit to global level:

- Provide detailed designs for the requirement developed in the current or next iteration.
- Redesign software developed in previous releases (a release is a set of several iterations).
- Provide overall product vision, to keep a global coherence throughout the project and developed software.

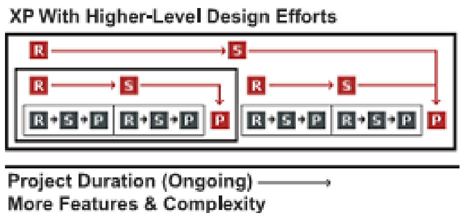

XP With Higher-Level Design Efforts
R
S
P
Project Duration (Ongoing)
More Features & Complexity

Figure 3. Parallel design tasks presented in [2].

This process covers the four UCD activities. The evaluation of designs against requirements is supported by the redesign activities and an overall product vision.

This process respects the following UCD principles:

- Understanding of users, tasks and environment: the focus is on the design in this process, that encloses the respect of this principle
- Iterative: the process is iterative on each level
- Multidisciplinary: involvement of designers.
- However it is not clear if the evaluation of the designs are driven and refined by users (third UCD principle) or if the users are involved (second UCD principle) but there is no counter-argument to respect these principles.

C. *Sequence of an iterative design phase and an iterative development phase*

Deuff et al. [8] present another proposition of process for Agile-UX that gives a good place to an upfront designing (see Figure 4.). They justify this iterative design phase by the fact that time is necessary before development to build the context (gather data on users, their tasks, the context, etc.) and make the first design propositions. But, the time is not available in classical Agile processes. So usability experts have to juggle between too much tasks (gather the necessary data, define the design, test) while trying to maintain a global vision during iterations. To resolve this issue they propose to cut the project in 3 phases: Design, Development and Final test. The design and the development phases are iterative. Even if a Final test is planned, several users' tests are done during the first and second phase.

This process covers the four UCD activities if the phase 1 is dedicated to understanding and specifying the context of use, specifying the user and organizational requirements and producing design solutions. But the description of the process is not deep enough to ensure that phase 1 covers these activities. The evaluation of designs against requirements is covered by Phase 3 and by the regular tests done throughout the project.

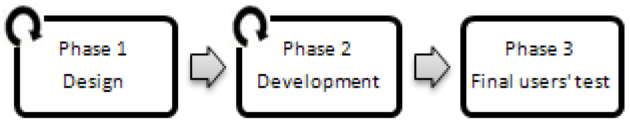


Figure 4. Deuff's process proposal [8].

This process respects the following UCD principles:

- Understanding of users, tasks and context: notably through phase 1 of designing
- Users' involvement: users are involved throughout the project in particular thanks to regular testing.
- Evaluation: design and software are iteratively evaluated by users and it is enhanced by phase 3 which plans a final users' test
- Multidisciplinary: designers and users involvement.

The fourth (iterative) UCD principle is more or less respected since the first and second phases are iterative but a global loop is missing.

D. *Big upfront design*

Agile methods do not encourage a big upfront design [4][14][15]. Or more precisely this upfront design is out of the scope of the Agile methods. In fact an analysis conducted by the product owner is necessary to define the product backlog, but no best practice is defined to support the product owner for this task, which is done before the start of the development Agile process. To support the product owner for this task, some usability experts propose to conduct a big analysis up front. Others are against this practice and prefer to use the iteration called "zero" to conduct a short analysis and then go deeper throughout the project according to the needs of analysis. Big upfront design in Agile-UX has supporters and opponents (see TABLE I.), their arguments are presented bellow.

1) Supporters of a big upfront design: Chamberlain [6] in his principle 4 for integration UCD and Agile development insists on a big upfront design before any development: "UCD practitioners must be given ample time in order to discover the basic needs of their users before any code gets released into the shared coding environment." This time is necessary to capture users' needs, usability goals, context of use and design criteria. It is also used to define users or to build personas. In some cases, at least a part of the designs is defined in this step which is not recommended by Nodder [14]. It's even risky according to Blomkvist [4] and Deuff [8] to engage a project in a development without this initial analysis and design. Agile methods are intensive during iterations, so that usability experts do not always have time to ask questions or to take a global view and ensure the homogeneity and consistency of the solution.

For Brown [5], long research projects are sometimes necessary to devote more time in analysis in order to gather the necessary data.

TABLE I. REPARTITION OF OPPONENTS AND SUPPORTERS OF A BIG UPFRONT DESIGN AND THEIR ARGUMENTS

		[2]	**[4]**	**[6] Prj. I**	**[7]**	**[5]**	**[8]**	**[11]**				**[14] Prj. PV**	**[15]**
								Prj. 1	***Prj. 2***	***Prj. 3***	***Prj. 4***		
Supporters	Do first analysis and design		X	X			X		X	X	X		X
	Avoid risks		X				X						
	Have a global vision		X		X	X	X					X	X
Opponents	Avoid risks (time & money consuming)	X		X		X		X					
	Respect Agile values: accept changes	X				X		X					X
	Big upfront analysis reduce quality	X											

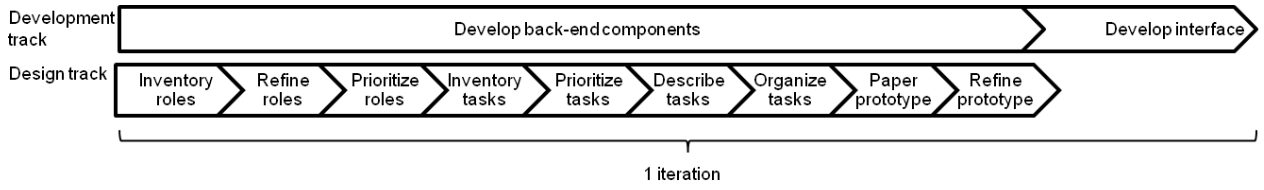


Figure 5. One iteration in Usage-Centered Design adapted to Agile methods [7].

2) *Opponents of a big upfront design:* In opposition, for Armitage [2] it is too risky and time and money-consuming to design deeply beforehand and it is totally against Agile practices which encourages “trial and error to reduce the risk of building the wrong thing”. A big upfront design might reduce quality of the software and its design [2]. Another problem is the difficulty to accept changes later when a big upfront design was done, which goes against the Agile values “Responding to change over following a plan” and “Working software over comprehensive documentation” [2][11][15]. For Brown [5], gathering data or needing design validation is not a justification for an upfront design phase: these tasks can be conducted throughout the project thanks to the planning of regular meetings with users. These meetings can serve to discuss all the elements already built (wireframes, personas, software, etc.) with users but also to gather data on their tasks etc.

3) *Conclusion:* For Brown [5], it is a myth that no upfront design is allowed in Agile-UX. In fact, Agile developers all work with a kind of high-level plan also called a roadmap. It is also necessary for usability experts to develop a kind of roadmap in the form of, e.g., a simple sketch, a workflow diagram, wireframes or Post-its. This way the team has to take the time to build this global vision while taking care not to spend too much time and fall into the track of a design phase that never ends. This is necessary to identify proactively technical impediments.

This process covers only three of the four UCD activities: understanding and specifying the context of use, specifying the user and organizational requirements, and producing design solutions. Furthermore it depends on the tasks done in this big upfront design phase: in fact the proposition of designs is not always included, sometimes it is diluted in the iterations following this first phase. So a big upfront design is not enough to ensure that designs will meet the users' requirements.

This process does not ensure the second (users' involvement), the third (evaluation), the fourth (iterative) and the sixth (multidisciplinary) UCD principles even if they are recommended to ensure a better design. In fact the goal of this process is to answer to the first UCD principle: understanding of users, tasks and context.

E. Usage centered design

Constantine [7] proposes another approach, which is the integration of Usage-Centered Design, and not User-Centered Design, and Agile (see Figure 5.).

Usage-Centered Design is more focused on roles than on users and on usage scenarios also knew as task cases. Roles and tasks are identified by stakeholders (domain experts, business people, designers, developers, users, etc.) thanks to brainstorming. The process is composed of iterations that are all composed of these succeeding steps: (1) Inventory roles; (2) Refining roles; (3) Prioritizing roles; (4) Inventory tasks; (5) Prioritizing tasks; (6) Describing tasks; (7) Organizing tasks; (8) Paper prototype; (9) Refining of prototype. During this time developers develop the back-end components. When the prototype is refined, they develop the interface.

This process covers only three of the four UCD activities: understanding and specifying the context of use, specifying the user and organizational requirements, and producing design solutions. The evaluation of designs against requirements is not covered; it goes against the third UCD principle [16].

As stakeholders are consulted to define roles and tasks, the second (users' involvement) and sixth (multidisciplinary) UCD principles are respected. The process is intrinsically iterative (principle 4.). The good definition of roles and tasks answers to the first UCD principle, even if the process does not ensure the understanding of the environment.

V. DISCUSSION

The facing of the presented Agile-UX processes to the UCD activities shows that they respect generally the four UCD activities (see TABLE II.). However the activity of evaluation is not covered by the big upfront design or by the Constantine's process: they have to be completed.

None of the presented processes ensures the fifth UCD principle (see TABLE II.), this principle aims to improve the whole user experience by addressing the support of users in their use of the product. This can be addressed by all processes if the willpower to care about it exists in the project and in the team. Sy's, Deuff's and Constantine's processes clearly involve users at least to support the evaluation of designs and software and/or to define the context and the needs (see TABLE II.). Armitage's and the big upfront design processes do not ensure this involvement and evaluation, but the respect of these UCD principles is recommended to improve the designing and the meeting of the users' needs (see TABLE II.). Sy's, Armitage's and Constantine's processes are strongly iterative (see TABLE II.). Deuff's process is more or less iterative, this is due to the introduction of an upfront analysis separated from the development phase, but each phase is iterative (see TABLE II.). For the big upfront process it is recommended to make it iterative but it is not ensured (see TABLE II.). Finally, all presented processes involve at least designers (see TABLE II.), and even if some of them do not ensure the involvement of developers or stakeholders, including the end-users, it is at least recommended.

Evaluate these Agile-UX processes under the Agile values is not an easy task. Firstly, as they are processes they can go instead of the first Agile value (Individual and interactions over processes and tools) (see TABLE II.). We can understand that processes promote a separate analysis and design phase, as Deuff's and big upfront design, are certainly more rigid and thus do not encourage the third Agile principle (Customer collaboration over contract negotiation) by fixing the designs before development and the discovery of impediments (see TABLE II.). As the separate design phase aims to produce designs, we can deduct that both processes do not promote the second (Working software over comprehensive documentation) (see TABLE II.). The more iterative attitude of the Sy's, Armitage's and Constantine's processes respects better the third Agile principle (see TABLE II.). Sy and Constantine both insist on the necessity to reduce documentation by doing designs (as paper prototypes) but only when it is essential for communication and exchange or to support specification of the user stories, in the respect of the second Agile value (see TABLE II.).

Finally, respecting the Agile values is more a question of attitude adopted by the team, a question of culture, that something intrinsic to the Agile-UX emerging processes.

TABLE II. AGILE-UX PROCESSES FACING TO UCD ACTIVITIES AND PRINCIPLES AND TO AGILE VALUES

		Sy's process	**Armitage's process**	**Deuff's process**	**Big upfront design**	**Constantine's process**
UCD Activities	1. Specify context	X	X	X	X	X
	2. Specify users' needs	X	X	X	X	X
	3. Design	X	X	X	X	X
	4. Evaluate	X	X	X	NO	NO
UCD principles	1. Design based on explicit understanding of users, tasks and environment	X	X	X	X	X
	2. Users involved	X	Not ensured	X	Not ensured but recommended	X
	3. Design driven and refined by user-centered evaluation	X	Not ensured	X	Not ensured but recommended	NO
	4. Iterative process	X	X	More or less	Not ensured but recommended	X
	5. Process addresses the whole user experience	Not ensured	Not ensured	Not ensured	Not ensured	Not ensured
	6. Team includes multidisciplinary skills	X	X	X	Not ensured but recommended	X
Agile Values	1. Individual and interactions over processes and tools	Not ensured	Not ensured	Not ensured	Not ensured	Not ensured
	2. Working software over comprehensive documentation	Not ensured but promoted	Not ensured	Not ensured	Not ensured	Not ensured but promoted
	3. Customer collaboration over contract negotiation	Not ensured	Not ensured	Not ensured	Not ensured	Not ensured
	4. Responding to change over following a plan	X	X	More or less	NO	X

VI. Conclusion and Future Work

Even if the parallel tracks process is generally accepted, some other processes are proposed. This echoes Brown [5] who explains that one myth of Agile-UX is to believe that there is only one way to do it. Every team has to find its proper way to process Agile-UX because “different challenges require different solutions”. This corresponds perfectly with Agile values, notably “Individuals and interactions over processes and tools”.

Following the analysis of the different Agile-UX processes proposed in literature, we can observe that no one covers entirely all the UCD activities, UCD principles and Agile values. To ensure the respect of all these principles, each analyzed process should be completed by practices or by cultural aspects. For instance Constantine’s process should be completed by tests. Armitage’s process works more on the global vision than the other processes, it may be associated with Sy’s process to improve it. Deuff’s process makes a major contribution on the organization of the tests (not detailed in this paper) but the separation of an analysis phase and a development phase are in contradiction with Agile that fights against upfront analysis and design phase by its fourth principle (Responding to change over following a plan). This analysis brings out questions to investigate in future work:

- Which practices are necessary to complete the Agile-UX processes?
- What can be an Agile-UX process that respects all UCD and Agile principles?
- How may the people and the cultural question enhance the Agile-UX processes?
- How to ensure the respect of the fifth UCD principle: process addresses the whole user experience?

Acknowledgment

Many thanks to my colleagues (in particular Sylvain Kubicki) who support my work on Agile-UX through exchanges and experimentations and those who have helped me in particular to write this paper: Muriel Foulonneau and Jocelyn Aubert.

References

[1] A. Alliance, Agile manifesto, 2001, Online at http://www. agilemanifesto.org, 08.01.2013.

[2] J. Armitage, Are Agile methods good for design?, interactions, 11(1), 2004, pp. 14-23.

[3] A. Bankston, Usability And User Interface Design In XP, 2003, White Paper, http://www.ccpace.com/resources/documents/usability inxp.pdf, 08/01/2013.

[4] S. Blomkvist, Towards a model for bridging Agile development and user-centered design, In Human-Centered Software Engineering—Integrating Usability in the Software Development Lifecycle, 2005, pp. 219-244, Springer Netherlands.

[5] D. D. Brown, Five Agile UX Myths, Journal of Usability Studies, 8(3), 2013, pp. 55-60.

[6] S. Chamberlain, H. Sharp, and N. Maiden, Towards a framework for integrating Agile development and user-centred design, In Extreme Programming and Agile Processes in Software Engineering, 2006, pp. 143-153, Springer Berlin Heidelberg.

[7] L. L. Constantine and L. Lockwood, Process agility and software usability: Toward lightweight usage-centered design, Information Age, 8(8), 2002, pp. 1-10.

[8] D. Deuff, M. Cosquer, and B. Foucault, Méthode centrée utilisateurs et développement Agile: une perspective «gagnant-gagnant» au service des projets de R&D, In Conference Internationale Francophone sur I'Interaction Homme-Machine. Sept. 2010, pp. 189-196, ACM.

[9] I. DIS, 9241-210: 2010, Ergonomics of human system interaction-Part 210: Human-centred design for interactive systems, 2009, International Organization for Standardization (ISO), Switzerland.

[10] Extreme Programming: a gentle introduction, http://www. extremeprogramming.org/, 08.01.2013.

[11] J. Ferreira, J. Noble, and R. Biddle, Up-front interaction design in Agile development, In Agile Processes in Software Engineering and Extreme Programming, 2007, pp. 9-16, Springer Berlin Heidelberg.

[12] Z. Hussain, W. Slany, and A. Holzinger, Current state of Agile user-centered design: A survey. In HCI and Usability for e-Inclusion, 2009, pp. 416-427. Springer Berlin Heidelberg.

[13] D. Kane, Finding a place for discount usability engineering in Agile development: throwing down the gauntlet, In Agile Development Conference, 2003, ADC 2003, Proceedings of the, Jun. 2003, pp. 40-46. IEEE.

[14] P. McInerney and F. Maurer, UCD in Agile projects: dream team or odd couple?, Interactions, 12(6), 2005, pp. 19-23.

[15] C. Nodder and J. Nielsen, Agile usability: best practices for user experience on Agile development projects, Nielsen Norman Group, 2010.

[16] A. Nummiaho, User-Centered Design and Extreme Programming, In Software Engineering Seminar, 2006, pp. 1-5.

[17] D. Rawsthorne and D. Shimp, Scrum In A Nutshell, SCRUM alliance, http://www.scrumalliance.org/articles/151-scrum-in-a-nut shell , 08.01.2013.

[18] A. Seffah, J. Gulliksen, and M. C. Desmarais, Human-Centered Software Engineering - Integrating Usability in the Software Development Lifecycle, 2005, p. 32, Springer

[19] D. Sy, Adapting usability investigations for Agile user-centered design, Journal of usability Studies, 2(3), 2007, pp. 112-132.

Distributed Agile Software Development Challenges and Mitigation Techniques: A Case Study

Abdullah Saad Alqahtani, John David Moore, David K Harrison, and Bruce M Wood
School of Engineering and Built Environment, Glasgow Caledonian University, Glasgow, United Kingdom
{abdullah.alqahtani, j.d.moore, d.harrison, b.wood} @gcu.ac.uk

***Abstract*—There is a growing interest in applying Agile development methods alongside global software development in order to reap the benefits of both approaches. With this said however, research has shown that software companies are encountering significant challenges when attempting this due to the contradiction between Agile values and the global development environment. This paper focuses on the challenges encountered with this kind of development and discusses several techniques via which these challenges can be addressed. It presents a case study and applies interviews with a software development company adopting the distributed Agile approach. From this study it can be seen that the communication barriers are the biggest development challenge. The development teams and product owners need to work hard to increase the level of communication between them by having a daily, regimented communication schedule. Flexibility with the working hours and location is an important practice with regards to limiting the barriers of the distributed development.**

Keywords-distributed Agile; global Agile; global software engineering; Agile software development.

I. INTRODUCTION

Increased globalization has led to greater competition between software development companies around the world. The software development industry is seeing a shift from co-located software development to Global Software Development (GSD), which involves multiple distributed development teams from different locations. GSD facilitates competitive software development prices by using teams from countries that have an abundance of IT developers available at relatively low cost. In addition, research has shown that software companies are interested in applying Agile Software Development (ASD) to develop the software by global teams to have the combined advantages of ASD and GSD [1][2]. The combination of Agile development methods and GSD is known as Distributed Agile Software Development (DASD). Venkatesh defined Distributed Agile Development as: "*Distributed Agile, as the name implies, is a model in which projects execute an Agile Methodology with teams that are distributed across multiple geographies*" [3]. This combination has shown signs of providing IT companies with the ability to meet the critical success factors of the software industry, such as quality, time, and cost. Sutherland et al. [4] detail their experience of applying a distributed Scrum approach and report several advantages such as the high increase of team productivity, an increase in the transparency between team members, better building of trust, and increased project visibility. However, although the potential advantages of GSD are clear, research has shown that software companies are encountering significant challenges by applying this approach. Developers are not always able to apply Agile practices successfully due to challenges introduced through the global development environment including distance and time zone differences [5].

This paper presents the results of a qualitative study involving a company which employs the DASD approach. The study focuses on the challenges of adopting the DASD and discusses some possible techniques to address and minimise those challenges.

This paper is structured as follows: first, the related work will be reported. Following this, the research method will be discussed and explained. Section III will describe the investigated company, before the strategy of development for the investigated company is reported. Results and discussion will be presented in Section VI, whilst the final section contains the summary and conclusion.

II. RELATED WORK

A systematic review studied applying Scrum practices in global software development using 27 literature studies and analyzed the challenges into three categories: communication, coordination, and control [6].

The challenges of using Agile with distributed national teams can be categorized into three types of lack: communication, trust, and control [2].

The effective communication within distributed Agile software development is a huge challenge. The reasons that create the communication challenges could be summarized into four categories: a lack of communication tools, time zone differences, a lack of English language, and a lack of teamwork. Those barriers may limit and decrease the communication in a distributed development [7].

There are current needs for more studies to understand how to adopt Agile methods with global software development. There is a lack of theoretical models of distributed Agile. More studies are needed to address the literature gap by investigating the geographical, cultural, and temporal challenges [8].

Previously, we conducted a systematic literature review focused on the challenges of applying DASD [9]. One of the significant findings of that review was that most of the DASD studies cover the technical perspective of the development and lack coverage of the human perspective. The review also reported that: "The human perspective needs to immediately search to explore the effect of the cultural differences on the relationship between the stakeholders and the development process" [9]. The present case study aims to address this issue by exploring the challenges and techniques of applying DASD from the developers' point of view (i.e., human perspective).

III. Research method

The research presented in this paper is from a single descriptive case study. Data was collected by structured interviews. The interviews were face-to-face and were recorded with a voice recorder. Also, notes of the main ideas and answers were taken during the interviews. The data was transcribed from verbal form to textual form. The transferred documents were then compared to the notes from the interviews, to ensure the reliability of the data. Following this, a thematic analysis was applied, which is an approach to identify the themes and patterns from the collected qualitative data [10], [11]. In addition, the data-driven method was selected for the thematic analysis of this study. The data-driven method regarding Asnawi can be summarized into five steps, as follows: *"(i) reducing the raw information, (ii) identifying themes within subsamples, (iii) comparing themes across subsamples, (iv) creating a code, and (v) determining the reliability of the code*" [12]. Finally, to ensure the validity and the reliability of the study's qualitative analysis and to identify any elements of bias by the researcher, two procedures were applied. Firstly, after the final code was developed, it was tested by other researchers, who applied it to the raw data to ensure that the code and theme analyses were correct. The second procedure was having the transcripts rigorously checked by other researchers, comparing them to the verbal records and the notes that had been taken. The aim was to identify any transcription errors or mistakes [12].

IV. The investigated company

The interviews were carried out at a large, global IT development company. The company has 27 offices distributed throughout 11 countries around the world: Australia, Brazil, Canada, China, Germany, India, Singapore, South Africa, Uganda, the United Kingdom and the United States. The company provides software design and delivery services, as well as development consulting services. It also produces customized software products as tools to support distributed Agile software development, thus helping the development teams to communicate, share information and track progress. The company applies Agile methods in order to develop its global software projects and has been involved in the software industry for the past 20 years. The company required to be anonymous within this study.

Three interviewees with good experience of Agile methods and the distributed development approach agreed to participate in this study. Participant-1 has experience working with more than 15 teams from the entire world covering the east and the west side including countries such as India, USA, UK, and Australia. Participant-2 has 4 years of experience including a special course in Agile development during his Master degree, and significant experience when it comes to with working with stakeholders from different cultures including people from China, Europe, UK, USA, and Middle East. Participant-3 acquired a vast amount of experience before joining this company as he developed a project while both the product owner and business analyst were away from the development team. He also has experience working with customers from different countries including New Zealand, Australia, and USA.

V. The development strategy

The investigated company applies a development strategy which goes through different stages before starting to develop the software. The first phase is the design phase. The project starts with meetings and the gathering of all participants in one place for a few days to finalize the requirements and estimate the deadline of the project. The inspection phase will come next where all the tasks should be broken into small stories. This involves the Project Manager (PM), the Quality Assurance (QA), the Business Analyst (BA) and software developers. The next stage is the analysis phase. The development stories need to be investigated during this stage to provide a better understanding of these tasks and create links between them. The BA plays a main role in this phase. The development then begins, by applying a weekly iteration to show the development case and update the other stakeholders. Each development team needs to have a daily meeting to track the development and identify any development issues.

VI. Results and discussion

The results of the thematic analysis classify the development challenges into four main themes: communications, cultural differences, management and control, and Agile skills

A. Communication and Collaboration Challenges

1) Lack of communication and losing the ability to make immediate decisions (A1): Agile methods require interactive, daily communication among stakeholders. This is difficult to provide within the global environment. The lack of communication and collaboration is a significant issue within the DASD approach [13]. Team members were not able to make immediate decisions, because of the distance between the participants and the lack of communication. As mentioned by Participant-3: "*We lose the ability to have an immediate decision. If we were here at 11am and we wanted to know something straightaway the earliest we could hear from our product owner will be 3pm and that's only if he's got up very early*."

2) Time zone differences (A2): The time zone differences is one of the main reasons that cause DASD's communication challenges [7]. The distance and time zone differences among stakeholders could reduce the available overlap of working hours of distributed teams. Participant-3 reported the issue of having no overlap of working hours by: "*I think if you had two teams where their working days didn't overlap at all, so if you had the UK and the East*

Coast of Australia where there's something like a 10 hour difference, I don't think that would work".

3) The lack of English language skills (A3): In most cases, the English language is not the mother tongue of the offshore team members. The lack of proficiency in English could pose a major challenge for the development teams. The different levels of English among the stakeholders could create misunderstandings [14], in the event of people trying to express or indicate meaning by a hint and expecting the others to understand them. Participant-3 reported that: "*If you're having a discussion and there's a thing that you don't say and you assume the other person knows and it's implied, that's where you get the chance for errors*".

Participant-2 who is not a native English Language speaker described his experience with communication with people with different level of English as hard. Participant-2 stated that: "*The other thing which might be hard is that different people have different levels of English knowledge.*". Also, Participant-2 mentioned some difficulties with understanding native speakers who are speaking with a difficult accent or speaking in a fast way: "*Sometimes it's hard to understand people who are speaking English as their mother language, as well*".

B. Communication and Collaboration Techniques

1) Find a time and a way for synchronised communication (B1): It is important to create an overlap of working hours among the distributed teams. The overlap hours will be used as available time for synchronised communication. Participant-3 reported there should be at least 2 hours of overlapping: "*If you have two teams in different time zones their working days have to have some overlap and if they don't have some overlap and if they don't have some overlap then you need to change the working hours of one of those teams so there is an overlap. I think there needs to be, I would say, at least two hours overlap between those two teams so they can talk face-to-face*".

With some cases that require staying late, the project manager and the business analyst could stay late to communicate with the other stakeholders. Participant-1 stated that: "*PM or BA or whoever needs to showcase something to the client, they need to stay for a while.*".

2) Flexibility regarding working from home (B2): Working hours should be flexible; therefore, the team members should be able to work from home when necessary. This flexibility could help to create overlapping hours among teams. Participant-1 reported that: "*Yeah so the company gives you the opportunity and flexibility to work from home. They also provided the broadband.* ". Participant-2 stated as well: "*The people are free to do and people are getting flexible times to do work from home or work from somewhere else when they are away from the office*".

3) The communication schedule should be regimented (B3): The development stakeholders should have a daily, regimented communication schedule. Such a schedule would help to increase the communication level. Participant-3 reported that: "*I think you need to do what we're doing here at this company and have a very regimented communication schedule*". The product owner should make himself available to communicate with the development team as Participant-3 said: "*I'd say from our product owner's point of view he's got to make sure that he's very involved and he keeps himself aware with what we're up to*". Communication, as reported earlier, is the main issue with the DASD development, so it is necessary to increase the level of communication among the distributed teams. Participant-3 summarised that by: "*You've got to make sure that you communicate well with the stakeholders*".

4) Ask people to speak clearly and be explicit (B4): Regarding the different levels of English skills among the stakeholders, there is a need to speak clearly and to be explicit about what is wanted. Participant-3 mentioned that: "*It's much better to be explicit and to really make clear what you want*".

5) Apply multi-channels for communication (B5): There is a need to have multi-channels for communication. There should be a choice of method and use of the one best suited, such as phone calls, video Skype calls, voice over IP and texting. Participant-2 reported that: "*We are using voice over IPs and the video services. We use Skype, we use GoToMeeting, we have an internal voice over IP device here*", and reported as well: "*we use our own internal service for chatting*". In addition, software to share the screen and knowledge helps teams to share information and increase the visibility of the development. Participant-2 mentioned that: "*So, I can say, tools are really important in distributed systems*".

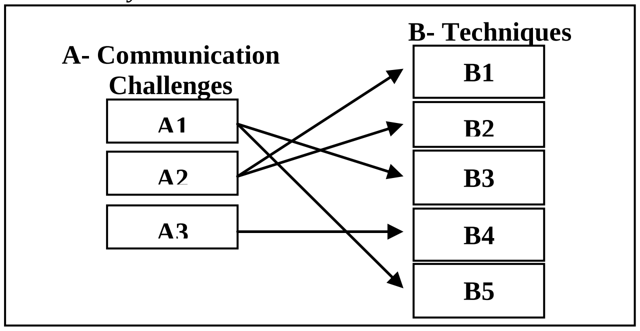


Figure 1. Communication with DASD challenges and techniques

Figure 1 illustrates the recommended techniques to address communication and collaboration challenges. It links the challenges with the techniques in order to provide better understanding of them. For example, to address challenge A1, techniques B3 and B5 can be employed.

C. *Cultural Differences Challenge*

The cultural differences of the stakeholders could create certain misunderstandings [14]. Participant-2 reported that: "*There are a lot of different things in a culture. Like, in some countries, people really like to talk about politics*". Cultural differences could limit the communication between development participants in order to avoid any misunderstandings. Participant-2 stated: "*I feel I know, if somebody from a different culture joins our team, how to behave and then how to find the limits on paid programming, how to speak to people, what sort of questions to ask, what sort of questions not to ask. So, these are the things which we learn*".

D. Techniques to Address the Cultural Differences

1) Creating an open culture within the development teams (D1): There is need to promote an open culture among the project's stakeholders, encouraging people to be free, flexible and liberal. Team members should accept other cultures and try to understand them. Participant-1 mentioned that: "*our culture rules are very liberal, free, there is no dress code. The people are free to do and people are getting flexible times to do work from home or work from somewhere else when are they away from the office. So this flexibility provides a lot of appreciation to the developers and all the people*". Participant-2 also stated that there is need to be flexible within the people from different cultures: "*people who are working in a distributed team, I guess should be more flexible than people who are working on a one - centralised process*".

2) Move the developers between the teams (D2): Providing the team members with the opportunity to move between global offices could help them to discover and explore other cultures. Participant-1 mentioned that by: "*there is a global assignment program which runs every year and it gives a chance to people to work round in any office in the world. So it's a very diverse culture in the company*".

3) A training course for new members (D3): New team members should have a special training course to provide them with the required Agile skills and make them aware of other cultures. The investigated company has a multi-cultural training centre in Bangalore, India. This could help new members to understand different cultures as reported by Participant-1: "*Once you hire anyone, if it's a fresh then he's a graduate. We send them to a university. There is a university which runs in India, in the Bangalore office*". And Participant-1 mentioned as well: "*All the students around the world gather with a different culture in India. They do works together on the same project for three months. After that we send them across different global assignments*".

4) Choose people who fit in with the distributed development culture (D4): Before hiring new people, they should be interviewed to ensure that they fit in with the open culture of the DASD. Participant-1 stated that: "*Always choose the people who actually fit with the culture. We don't choose people who don't fit with the culture*".

In addition, new members should have a qualifying period of a few months, to make sure they fit in with the development culture and environment as reported by Participant-1: "*Even after that, there is a probation of three months, okay. So in the three months itself it is enough time to know the person's attitude and whether - how he is behaving in all the steps. So if he doesn't fit in the culture then we don't extend their assignment*".

5) Flexible working hours and places (D5): This practice was mentioned when addressing communication issues and could also help to increase trust between the company and its employees, one of the cultural issues within the DASD. Participant-1 stated that: "*They do - they know all right that the company the flexibilities providing to them it come with a trust. So the company's putting trust on them so they, of course, need to do the work properly and they also need to put the trust in the company*".

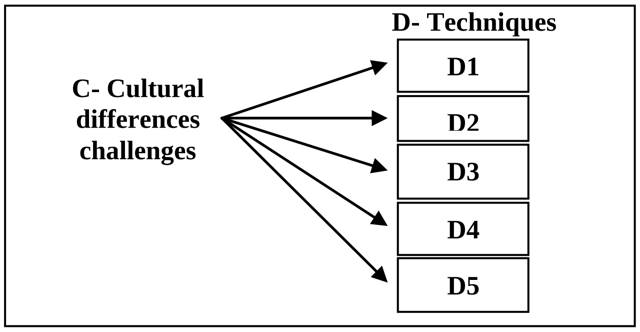


Figure 2. Cultural difrencess with DASD challenges and techniques

Figure 2 illustrates the recommended techniques to address the cultural differences challenges. Techniques D1 to D5 have been applied by the company to minimize the impact of the cultural differences to the development. The cultural differences could reduce the communication as reported early within this section and limit the collaboration between the team members.

E. Management and Control Challenges

1) Updating the developed story on the online wall (E1): Development participants with the DASD approach usually apply an online story wall to track progress. In some cases, they have issues with not updating the developed story on the online wall. This could lead to duplication when developing the required functions/stories. Participant-2 declared that: "*So, sometimes, you - when you get into a story and then it finishes the phase and you start another story, you may forget to move it on the electronic wall*".

2) Estimation difficulties (E2): The second management challenge is with estimation. Large teams could have difficulties with estimating their stories. Participant-2 explained this issue by: "*Estimation for example is one*

thing that it's hard. So when you have 20 people online and you have 20 people here and you want to estimate stories!!".

F. Management and Control Techniques

1) Increase communication (F1): There is a need to increase the level of communication in order to manage the work and to resolve any misunderstandings. Participant-3 mentioned that the communication is required to better apply DASD: "*If we do a lot of communication then we can apply all the practice of Agile globally*". Participant-2 reported that as well: "*There should be a lot of communications between the teams as well*". In addition, Participant-1 stated the same thing to manage the distributed Agile development: "*Any company you go there would be the challenge to manage such a vast distributed work, right? It requires a lot of communications; it requires a lot of co-ordinations between all the offices to work together right*".

2) Use software management tools (F2): Using software management tools is required to apply different Agile practices within the distributed development environment. Those tools support the development, make it more visible and easier to track. The tools usually have an online wall for the development stories, which is required to keep it coordinated with the normal story wall. Participant-3 declared that: "*So we have story wall, but that's all replicated in an online tool and we make sure that we keep those two in sync so that the product owner at any time can look at our entire story wall and see what's in progress*". And Participant-2 mentioned the same as well: "*is really important and there are not - and you should be able to first of all, be responsible for updating the electronic wall*".

3) Split large teams (F3): Having a large number of participants could make it difficult to apply some Agile practices, such as estimation. Therefore, splitting large teams could be a solution. However, this requires a lot of communication and coordination between the divided teams. Participant-2 agreed with that by: "*You split the team, you have two PM, you split the number of developers, you will add a new BA. But there should be a lot of communications between the teams as well*".

4) Estimation cards (F4): This practice aims to address the estimation issue. The participants would have a card to estimate each story and they would then show their cards and discuss issues. Participant-2 mentioned that technique by: "*we talk about the story and we count to three and everybody should show a card, or show their hands*".

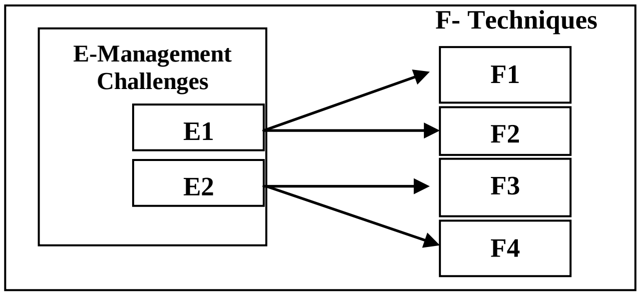


Figure 3. Mmanagment with DASD challenges and techniques

Figure 3 links the management challenges with the recommended techniques in order to provide better understanding of them.

G. Agile Level Challenges

1) Lack of a close relationship (G1): The distributed development could result in losing the main aspect of Agile, which is the close relationship between the development participants. Participant-3 mentioned that by: "*I think the main problem with global is - with Agile it's very important to maintain a close relationship to your customers*".

2) Working with traditional organisations/ customers (G2): Traditional organisations/customers may not accept the Agile way of development. They may be used to traditional development approaches, such as the waterfall model [4]. This could decrease the Agility level of the development. For instance, traditional organisations may take their time to allow the developers access to their database or to the necessary information. Participant-2 reported that: "*We speak a lot with tech team, with manager's team, with anyone who can - but they are traditional companies. They have a lot of paperwork for just getting one server, access to one server, or access to a database. But, in an agile company you just ask for something. In our company if you need to access anything…we just ask and we get it as soon as we can. But it's sometimes in other, in client side, in the companies which we are working for they have their own database team which we - a manager should give you permission*". And Participant-2 stated that as well: "*There have been problems with those things. Like database is the obvious one that we can say, you don't get the access to them. You need to go through their process*".

3) Difficulty in applying some Agile practices (G3): The global development setting could make it difficult to apply some Agile practices [14]. For example, the stand up daily meeting is difficult within the distributed Agile development, because of the large number of participants and the lack of visibility among the meeting attendees. Participant-2 reported that by: "*I guess the whole point of stand up is visibility so that you can see somebody and you can ask a question*", and by: "*So imagine if 100 people want*

to talk for one minute each, it would be a bout two hours while people are standing".

Furthermore, applying the retrospective practice with the distributed development is difficult as well. Participant-2 stated that: "*Retrospectives are getting affected. Because retrospectives in an agile team are, I guess I feel it's the most physical thing happens because what we do is that we practice different type of RETROS. So what we do is that every iteration that we have RETROS we change them. So we try a lot - because we don't want to make it boring*"

H. Techniques for the Agility Level

1) Use software tools to enable some Agile practices (H1): Usually, development teams adopt various software tools to help them to apply Agile practices. Participant-3 reported that: "*We've done some remote pair programming with him. We use tmux which is a UNIX tool for sharing terminals and we used a VNC client called Chicken and we also use Skype and SSH to set up the connection. So with a combination of those we can have a live pair programming session and that worked quite well*". In addition, Participant-2 stated that as well: "*Tools are really important,, learning how to work with tools are taking time. You may need more efforts*".

2) Dealing with the issues of traditional organisations (H2): Sometimes, IT development companies avoid working with a traditional product owner who is not able to understand Agile values. Sometimes, they try to provide the traditional product owner with some training about the Agile approach before the project begins. Participant-2 mentioned that: "*So the way that we work is that we try not to accept projects in our company that clients don't give us the chance of working in a way that we want. But some projects it happens that we try - so in some projects when the clients accept that we work for them, but they are not working in agile way. So usually we try to teach, teach the team which we are going to work with them. We communicate a lot, we talk a lot, we have lots of meetings in our team. So we try to settle these things before accepting a project*".

3) Practice for the stand up daily meeting (H3): Practice includes throwing a ball during the meeting. The member who has the ball is the one who is allowed to speak. This practice aims to manage the meeting by allowing one person to speak at a time. In addition, they hold computer tablets, such as iPads, during the meeting to see the distributed members. This practice reported by Participant-2 as: "*we use iPad and we ask them to be online and they talk about it. So we have a ball as a token. We throw it to each other when someone is going to talk.*".

4) Apply simple documentation (H4): One of the techniques in the DASD approach is doing simple reports to share information from the meetings with participants who were not able to attend. Participant-2 declared that by: "*Usually one person writes a simplify - a very simple report that this happens, this decision has been made. This is the reason that we make this decision. So we just read that email every night for example and we get updated about what's happening. If we don't like it, we can state it the day after, or we can send an email and discuss it*".

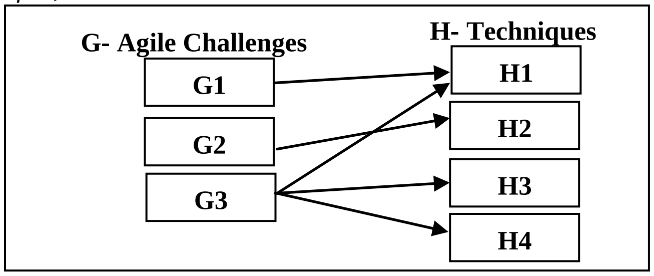


Figure 4. Agile challenges and techniques with DASD

Figure 4 reports the Agile challenges and links them with the recommended techniques to award better apply for Agile methods with the distributed development.

VII. SUMMARY AND CONCLUSION

The reported results suggest that communication barriers are the biggest challenge faced when employing the DASD approach. A number of techniques were reported by the participants to address the known communication issues with this approach. Most of the issues related to the lack of communication between stakeholders. The development teams and product owners need to work hard to increase the level of the communication between them.

The other main issue was the lack of Agile skills and knowledge from the developers and the product owners. The global setting makes this issue more clear because of the distance between the stakeholders. There is need to improve the Agile knowledge by applying training courses and Agile coaching to ensure the sufficient application of Agile practices.

The management issues are also related to the distance and the size of the development teams. Improving the communication level and Agile skills could reduce the management difficulties. Splitting the team may be applied with teams which have a large number of developers.

The issue of cultural differences is the least important problem because most of the stakeholders are aware of the other cultures and have the ability to work with different people. However, some misunderstanding could arise, particularly with the lack of communication. Thus, it is essential that the development participants are clear, flexible, and open with other cultures. The experience with DASD from the investigated company helped to understand the cultural differences challenges. The applied techniques such as training courses help to minimize the cultural differences issues. Moving the team members around the development teams throughout the world will help them to better understand the other cultures and could address this issue.

In conclusion, this case study highlighted some of the major challenges of applying DASD. It has also listed development practices to award a more effective application of this development approach. The discussion showed that the study findings are in agreement with existing literature for most of the investigated points.

Future work will involve further investigation in order to develop a better understanding and guidance towards applying Agile practices within a global setting.

VIII. ACKNOWLEDGMENT

This work is supported by the University of Dammam, the Ministry of Higher Education, Kingdom of Saudi Arabia, (PhD scholarship).

REFERENCES

[1] M. Nisat and T. Hameed, “Agile methods handling offshore software development issues,” 8th International: Multitopic Conference, Proceedings of INMIC, 2004, pp. 417-422.

[2] A. Szőke, "Optimized feature distribution in distributed agile environments," Product-focused software process improvement, 2010, pp. 62-76.

[3] U. Venkatesh, Distributed Agile: DH2A - The proven Agile software development approach and toolkit for geographically dispersed teams. New Jersey: Technics publications LLC., 2011.

[4] J. Sutherland, G. Schoonheim, N. Kumar, V. Pandey, and S. Vishal, "Fully distributed scrum: Linear scalability of production between San Francisco and India," Agile conference, IEEE, 2009, pp. 277-344.

[5] E. Hossain, M. Babar, and H. Paik, “Using Scrum in global software development: A systematic literature review,” Fourth IEEE International Conference on Global Software Engineering, 2009, pp.175-184.

[6] E. Hossain, P. Bannerman, and D. Jeffery, "Scrum practices in global software development: A research framework," Product-focused software process improvement, pp. 88-102., 2011.

[7] S. Dorairaj, J. Noble, and P. Malik, “Bridging cultural differences: A grounded theory perspective,” Proceedings of the 4th India Software Engineering Conference, ACM, 2011, pp. 3-10.

[8] D. Smite, N.B. Moe, and P.J. Agerfalk, "Agility Across Time and Space: Summing up and Planning for the Future," Agility Across Time and Space. Springer Berlin Heidelberg, 2010, pp. 333-337.

[9] A.S. Alqahtani, J. Moore, D. Harrison, and B. Wood, “The challenges of applying distributed Agile software development: A systematic review,” International Journal of Advances in Engineering & Technology, Vol. 5, Issue 2, 2013, pp. 23-36.

[10] R. Boyatzis, Transforming qualitative information: Thematic analysis and code development. SAGE publications incorporated. 1998.

[11] C. Dawson, Introduction to research methods: A practical guide for anyone undertaking a research project. Oxford: How To Books Ltd. 2009.

[12] A. Asnawi, A. Gravell, and G. Wills, “Emergence of Agile methods: Perceptions from software practitioners in Malaysia,” AGILE India, 2012, pp. 30-39.

[13] M. Paasivaara and C. Lassenius, “Using Scrum Practices in GSD Projects,” Agility Across Time and Space. Springer Berlin Heidelberg, 2010. pp. 259-278.

[14] M. Kajko-Mattsson, G. Azizyan, and M.K. Magarian, “Classes of distributed agile development problems,” The Agile 2010 Conference, IEEE, 2010, pp. 51-58.

Agile-User Experience Design: With or Without a Usability Expert in the Team?

Lou Schwartz
Public Research Centre Henri Tudor
Luxembourg, Luxembourg
lou.schwartz@tudor.lu

***Abstract*— In the past decade, numerous experiments of Agile-User Experience Design (also called Agile-UX) have been carried out. Through these experiments it remains unclear who should be in charge of the usability in an Agile-UX project development. After a review of the literature about the involvement of usability expert(s) in Agile-UX, this paper repeats two experiments which explore the necessity to involve usability experts in the team. The first experiment is based on the statement that developers should be able to manage the User-Centred Design (UCD) and conduct the related methods without the intervention of a usability expert, in order to respect agile practices. The second one is based on the statement that integration of a usability expert in project teams ensures better implementation of UCD and better results. Results of both experiments are discussed to validate research hypotheses for future work.**

Keywords- Agile-UX; team composition; use case

I. INTRODUCTION

Agile-UX is a project management principle for software development based on the Agile values and principles in respect to User-Centred Design (UCD) and supported by UCD good practices and methods. Nowadays, no official definition of Agile-UX exists, but a lot of experiments demonstrate its value [2][3][4][5][7][8][9][10][12]. In the literature, Agile-UX is implemented with the involvement of usability expert(s) in the Agile process and the use of methods from UCD. But, in Agile principles, intervention of experts is not encouraged [5]: dissemination of skills is preferred to the intervention of experts. We test both approaches through two qualitative experiments. The first one fully respects the principles of Agile project management: developers should be able to manage themselves, UCD and conduct the related methods without the intervention of a usability expert. The second option integrates a usability expert in the project team to ensure better implementation of UCD and better results. Results of both experiments are discussed to elicit future research questions for future work.

After a review of the literature on the involvement of usability experts in an Agile-UX development process in Section II, the paper will present two qualitative experiments in order to validate the relevancy of our hypotheses in Section III. Then the suitability of our hypotheses will be discussed in regard to the experiments' results in Section IV.

II. USABILITY EXPERT(S) INVOLVEMENT IN AGILE-UX

Though numerous experiments of Agile-UX, the question of "who is in charge of UCD" often comes [2][3][4][5][6][7][9][10][12]. Different options are exposed, but they are often the same, which we can regroup in 4 categories as explained below.

A. One usability expert

Only a couple of experiments advocate the integration of only one person in charge of UCD in Agile-UX ([4] project 1 & 3, [5] project PV, [9]). Often in this case, the UCD designer is also the product owner (project 1[4]) or developer (project 3 [4]).

B. A parallel team of several usability experts

In most cases, a parallel team of several usability experts is dedicated to the project ([2][3], Project 2 [4], [6][7][12]).

But, they organise the exchanges and work between developers and designers differently. In Agile methods, it is possible to dedicate a spike (an iteration to focus on a particular problem like test a new technology) to usability exploration. But, it is not a good solution to maintain a constant pace [7]. Some projects also involved occasionally UCD experts on some particular points (projects MG & PV in [5]); this is close to an organisation by spikes. But, for McInerney [5], it is important that the usability expert is available "on call" at all times, which may be impossible if the usability expert works on several projects simultaneously. Some other projects integrate usability in the iteration without real planning (see [P3.290] in [4]).

Sy [12] proposed a parallel tracks organisation of work: designers work with one or two iterations ahead of developers. To implement this proposition, several usability experts are needed, because of the amount of work and to respect best practices, which recommends that it should not be the same person who designs and evaluates the developed software.

C. UCD expert as product owner

In regards to the UCD expert's responsibilities and product owner's responsibilities, it is sometimes preferable to merge both roles (Project 1 [4], Project TB [5], for Beck in [6][10][12]). The product owner has the following responsibilities:

- Define the features of the product, decides on release date and content [11]. In this case, a UCD expert will be based on the gathered data of the users, on context and on tasks in order to define the user stories to develop [10].
- Be responsible for the profitability of the product (ROI) [11]: for this, the usability expert goes by the

context studies and the exchange with the organisation on the needs and the attempted profitability.
- Prioritize features according to market value [11]: this prioritization is done thanks to the exchanges with users and developers [10].
- Can change features and priorities every 30 days [11]: UCD expert accepts changes and modifies designs when it is necessary. He can modify user stories and prioritization according to new analysis.
- Accept or reject work results [11]: through the users' tests, expert validations and participation to the acceptance tests writing.
- Negotiate with all stakeholders [10].
- Communicate with the users and train the users [10].

Furthermore, some observations show that the product owner is often submerged by the marketing and sales concerns. He often does not have the skills to manage a user-centered design, and, as a consequence, he may lose focus on a user experience vision [10].

Sometimes, two product owners are appointed: one as the usability product owner and the other as the conventional product owner [10]. In this case, they commonly specify the needs and prioritize the work to do. This is an answer to some observations concluding that usability tasks are often not a priority because working software is still preferred to usable software and usable software is more expensive in terms of efforts and time.

D. *Team member(s) as responsible of the UCD process*

The last possibility explored is to take on the responsibility of the UCD process. It is also the more closed one of the Agile vision: do not involve a usability expert, but give this responsibility to one or more team members (Project 3 [4] & in part Project PV [5]).

In all these experiments, usability experts are involved in the Agile-UX projects. But in Agile principles, intervention of experts is not encouraged [5]. This can raise the following question: is it necessary to involve usability expert(s) in the team or is involving team members with some knowledge on usability sufficient? This is what we tested in the implemented experiments.

III. EXPERIMENTS

After the literature review and several exchanges with Agile professionals, we focused on the question of the usability expert involved in the team. We propose the following statements to test:
- S1: without usability expert, if the project team has sensitivity and some knowledge in HCI, Agile-UX works.
- S2: with usability expert involved in the project team, usability of the produced product is better than in S1.
- S3: the dynamic of the project team is better when a usability expert is involved.

We retrospectively and qualitatively question these statements through two experiments. We focus only on the usability of the final product, laying aside any potential cost overhead induced by the involvement of a UCD expert.

A. *Context of the experiments*

The method used consists of a retrospective and qualitative analysis of two experiments that tested two versions: the first, without a usability expert in the team (S1, S3), the second one, with one UCD expert in the team (S2, S3). The observations made will help us to better define the issues related to "who should play the role of the usability expert in Agile-UX?"

Both experiments are instantiations of Agile-UX and aim to develop a mobile application prototype, in order to demonstrate the interest of mobile touch-based applications for construction site-related activities.

The implemented prototype allows taking photos localized by Global Positioning System (GPS) on a construction site. The user can highlight parts of a photo (e.g., add an arrow to the default on a wall) and add textual or vocal notes about the entire photo or about the highlighted parts on the photo. The user can also register some construction sites by indicating their localisation on a map. Then the photos are automatically attached to a construction site according to their localisation. The user can also find his photos in his calendar since the photos are automatically attached to events in his Google® calendar based on the shooting date. Finally the user can share a set of photos with additional comments.

Two phases of development were planned to experiment two different implementations of Agile-UX. We have chosen Scrum as Agile method for both.

B. *Case #1 – Agile-UX without UCD expert*

1) Statement and composition of the team: In the first experiment, the team was composed of a full-time developer, a Scrum master (part-time), and a business expert (part-time) who plays the role of product owner, researcher and architect, with knowledge of architects' practices in France and Luxembourg.

All members of the team are sensitive to and have some knowledge in Human Computer Interaction (HCI). We have voluntary not involved a usability expert to test this configuration, which is the more suitable with the principles defined in Agile. Indeed, in Agile teams, everyone should be able to work on each part of the software development. So, after a while, team members should have sufficient knowledge and skills to relieve other team members of their tasks including, in case of Agile-UX, on usability tasks.

2) Implementation of the UCD: The first experiment lasted six months with iterations' duration of one week. We implemented Agile-UX on 22 iterations. The developer implemented only three usability methods: wireframing, users' tests, and satisfaction questionnaire.

3) Methods used

- Brainstorming sessions to build the product backlog including business experts and technical experts
- Wireframing with Microsoft Power Point®
- Two user tests:
  - Real situation of use, one user, one week
  - 6 architects, 6 scenarios, observation tests

4) Team dynamics and satisfaction: During this experiment, the developer played the role of designer, developer and evaluator of the application. As the developer had to play these three roles, he had the feeling to progress slowly. Moreover, it is not easy to evaluate own work and to always question it.

The skills in HCI of all team members allowed avoiding some usability mistakes. But, as the tests results showed, a lot of usability issues were identified by users. Regarding these experiment results, Agile-UX works without a usability expert and with a project having some sensitivity and knowledge in HCI. This justifies our first statement S1.

It should be noted that the team was in constant contact with the product owner thanks to his presence at each specification meeting, each demonstration meeting, and during some stand up meetings. The product owner was also available to answer any team member's questions.

C. Case #2 - Agile-UX involving a usability expert

1) Statement and composition of the team: During the second experiment, the team was composed of a full-time usability expert, a full-time developer, a business expert (part-time) as product owner, and a Scrum master (part-time). The business expert and the Scrum master are the same as in the first experiment. The developer has neither particular sensitivity nor knowledge in HCI.

The focus is to develop, for the same mobile application, interoperability aspects with a collaboration platform dedicated to the construction sector, photo tagging and a search engine based on photo metadata.

2) Organisation of work and process: This development lasted six months with iterations' duration of two weeks. The developer began one month before the usability expert, because of calendar constraints, to first work on technical requirements. For independent reasons, the usability expert quit the project before the end of the six months. We only really worked two and half months with the complete team. The process followed was the parallel tracks proposed by Sy [12].

3) Methods used

- Brainstorming sessions to build the first version of the product backlog including business experts and technical experts
- Personas, that help to improve the product backlog
- Wireframing
- Expert review based on ergonomics criteria after each release
- User tests with four users: two who know the application, two novices
- Focus groups to evaluate wireframing.

4) Team dynamics and satisfaction: During this experiment, the usability expert played the role of designer and evaluator of the application. The whole team had the feeling to quickly progress and to go deeper in the functionalities proposed but also in the quality of the application. Furthermore, more methods of UCD were used and they were adapted differently. The test results showed a lower number of usability issues identified by users thanks to the integration of the usability expert and they are less critical. That justifies our second statement S2: Agile-UX provides better results with the involvement of a usability expert.

Moreover, we observe the natural instauration of a "pair designing" [8]: when developer was implementing wireframes, he sometimes asked the usability expert to join him and to explain and validate developed interfaces during the implementation; when the usability expert designed wireframes, she sometimes asked the developer to join her and to validate feasibility of wireframes during their design. Even if the developer had no skill in HCI at the beginning, he learnt the good practices throughout the project and quickly integrated them.

Furthermore, the team was in constant contact with the product owner by his presence during the specification meeting at the beginning of the iterations', the demonstration meeting at the end of the iterations', during some stand up meetings and his availability throughout the project to answer all emerging questions.

IV. Discussion

Since the results are only based on two experiments, hypotheses cannot be formally validated. Then, in the following section, only the suitability of the hypotheses for future research will be discussed.

S1 and S2 are justified by the satisfaction of users, which is "correct" in the first experiment and which is better in the second one (see Table I and Table II).

TABLE I. USERS' TESTS RESULTS IN THE BOTH EXPERIMENTS

		Number of problems meet	
		Use case 1	***Use case 2***
By importance of the problems (importance = number of testers who met the problem * seriousness of the problem)	1	5	2
	2	2	1
	3	3	1
	4	0	1
	6	1	1
	8	0	1
	10	1	0
	12	1	0
	15	1	0
	20	1	0
	TOTAL	15	7

TABLE II. USERS' SATISFACTION RESULTS

Percentage of users' satisfaction	Use case 1	Use case 2
Average	75,42 %	81.25%
Min	62.5 %	75%
Max	90 %	92.5%

TABLE III. COMPARATIVE TABLE OF THE BOTH EXPERIMENTS

		Use case 1	Use case 2
Team	Developer	1 full-time	1 full-time
	Scrum master	1 part-time	1 part-time
	Product owner	1 part-time, business expert	1 part-time, business expert
	Usability expert		1 full-time
	Sensitivity to UCD	All team members	All team members, except the developer
Organisation of work	Duration	6 months	Expected 6 months – in reality 2,5 months
	Iteration duration	1 week	2 weeks
	Number of iterations	22	5
	Process	Scrum	Scrum + Sy's parallel tracks
UCD methods	Wire framing	Power Point®	Paper and pen Balsamiq®
	Users' tests in direct observation	6 users, 6 scenarios	At every iteration end with 2 users who know the application and 2 novices
	Users' tests in real situation	1 user during 1 week	NO
	Satisfaction questionnaire	X	X
	Personas	NO	X
	Expert review	NO	X
	Focus groups		To evaluate the wireframes
Other methods used	Brainstorming	To build the product backlog	To build the product backlog
Team dynamic and satisfaction	Feelings of the team	Slow progression	• Quick progression • Go deeper in the functionalities proposed • Improve quality of the application
	Observed team dynamic	• No real dynamic • Demotivation	• Pair-designing • Developer increased his HCI skills
Results		Lot of usability issues but working software.	Lower number of usability issues identified by users and they are less critical. Better users' satisfaction And working software.

Without involving a usability expert we observe a discouragement and disincentive particularly of the developer. On the contrary, involving a usability expert helps maintain a constant pace in the team ([1], principle 8). No difference has been observed on the constant customer collaboration ([1], value 3). Some best practices emerged like "pair-designing" and the whole team improved their practices and knowledge concerning HCI (see Table III for a resume of both experiments). This could justify our third statement S3: the dynamic in the project team is better with a usability expert involved in Agile-UX.

However, the fact that in the first experiment, the team was composed of only one person (the developer) may be of influence. Indeed in the second experiment the team was composed of two persons (the usability expert and the developer), then the dynamic observed may be due to the edge effect of the number of people in the team.

V. CONCLUSION AND FUTURE WORK

These experiments addressed two kinds of Agile-UX implementations. Thanks to these experiments, we can validate that the initial statements are justified hypotheses for further studies. The next step is now to define protocols to validate these hypotheses.

Another possible implementation that Agile evangelists begin to propose is to place the usability expert as the product owner. Indeed, the product owner is responsible for the contact with users, the definition of needs and the validation of the work done. A priori, the usability expert and the product owner have part of their high level responsibilities which overlap. A future work will be to check the legitimacy of the following statement: "usability expert could play the role of product owner".

ACKNOWLEDGMENT

The author thanks members of the CRAI laboratory of Nancy and the project's extended team members: Sylvain Kubicki, Annie Guerriero, Fabrice Absil, Marion Zéler, Luc Caffard, Alain Vagner and Charles Gilbertz. This paper is dedicated to Marion Zéler.

REFERENCES

[1] A. Alliance, "Agile manifesto", 2001, http://www. agilemanifesto.org, 08.14.2013.

[2] J. Armitage, "Are Agile methods good for design?", Interactions, vol. 11, no 1, 2004, pp. 14-23.

[3] S. Chamberlain, H. Sharp, and N. Maiden, "Towards a framework for integrating Agile development and user-centred design", In Extreme Programming and Agile Processes in Software Engineering, Springer Berlin Heidelberg , Oulu, Finland, June 2006, pp. 143-153.

[4] J. Ferreira, J. Noble, and R. Biddle, "Agile development iterations and UI design", In Agile Conference (AGILE), Washington, DC, August 2007, pp. 50-58, IEEE.

[5] P. McInerney and F. Maurer, "UCD in Agile projects: dream team or odd couple?", Interactions, vol. 12, no. 6, 2005, pp. 19-23.

[6] E. Nelson, "Extreme programming vs. interaction design", FTP Online, 2002.

[7] C. Nodder and J. Nielsen, "Agile usability: best practices for user experience on Agile development projects", Nielsen Norman Group, 2010.

[8] A. Nummiaho, "User-Centered Design and Extreme Programming", In Software Engineering Seminar, Fall 2006, pp. 1-5.

[9] L. Schwartz, A. Vagner, S. Kubicki, and T. Altenburger, "Feedback on the definition and design of innovative mobile services", InProceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Service, ACM, Luxembourg, Luxembourg, August 2011, pp. 525-528.

[10] M. Singh, "U-SCRUM: An agile methodology for promoting usability", In Agile, 2008. AGILE'08, Conference, IEEE, Tonronto, ON, August 2011, pp. 555-560, IEEE.

[11] J. Sutherland and K. Schwaber, Scrum, "The scrum papers: Nuts, bolts, and origins of an agile process", 2007.

[12] D. Sy, "Adapting usability investigations for Agile user-centered design", Journal of usability Studies, vol. 2, no. 3, 2007, pp. 112-132.

Do Agile Principles and Practices Support the Well-being at Work of Agile Team Members?

Marja Känsälä and Seppo Tuomivaara
Innovations, Management and Knowledge Creation
Finnish Institute of Occupational Health
Topeliuksenkatu 41 a A, 00250 Helsinki, Finland
marja.kansala@ttl.fi; seppo.tuomivaara@ttl.fi

***Abstract*—In this work-in-progress paper, a preliminary review on the literature of the connections between agile methods and well-being at work is done. The viewpoint of well-being at work is important when considering agile software methodologies and techniques. A stage for an empirical research setting on these issues is also set. The research setting targets to inspecting how applying agile practices are experienced and features of agile methods that enhance and challenge well-being at work, i.e., what kind of implications agile methods are perceived to have for well-being at work. Well-being at work is studied from three different points of view: avoiding excessive strain, feeling of autonomous and meaningful work, and development and change at work. A holistic measure of well-being at work, applying agile practices and managerial implications will be developed further in the empirical research.**

Keywords: agile methods; teamwork; well-being; change

I. Introduction

One rationale behind agile methods is the need to increase the quality of systems development projects. Typical problems relate to timetable, budget, customer needs and market demands, communication and cooperation, and the level of competence. The problems also include different practices of the customer and the developer team, e.g., following waterfall methods vs. agile practices. The software crisis has meant fundamental need for a new paradigm: the need to respond to constant changes [1]. The newest solution to this has been agile methods. Agile methodologies and practices emerged as an explicit attempt to more formally embrace higher rates of requirements change [27]. From the developers' point of view agility means the ability to act according to changing customer needs and continuous change. It enables the project to advance systematically even when stable and perfect product planning cannot be done at the beginning of the project. This aims at higher quality and customer satisfaction.

Technological solutions do not solve all the problems related to software development. So, it makes sense to explore other factors related to quality, like project management and organizing of work. Despite agile methods are widespread there has been relatively little scientific research on their application and organizational gains [4][17]. Agile methods have often been studied from the point of view of system productivity and efficiency, but well-being at work has not been studied that much systematically. Agile principles hold many promises in relation to well-being at work in theory but there has been little scientific research on how they are applied in practice [16][22]. There has been research on, e.g., around agile methods and teamwork aspect, though. Indeed, in their systematic review Dybå and Dingsøyr [8] found human and social factors as one thematic group of agile literature.

The objective of this preliminary literature review is to inspect the connections between agile work practices and well-being at work and based on this provides a case research setting. Through this research setting a model for adopting practices that enhance the well-being at work in agile environment can be developed. The questions of this preliminary literature review are: 1) Do agile working practices support the well-being of agile team members and if so, how; and 2) What kind of challenges agile practices bring to maintaining well-being at work and sustainable productive work.

Next, in Section 2 A., our view of agile methods is presented. After that, in Sections 2 B and 2 C., the frame of well-being at work in planning and implementation phases of agile practices is presented. In Section 3, the case research setting is presented and in Section 4, the future work is described.

II. Theoretical Framework

A. *Agile Principles and Practices*

Four agile values have been stated in the Manifesto for Agile Software Development [29]. The most important one related to the well-being at work is valuing individuals and interaction over processes and tools. Principles behind the Agile Manifesto include the values broken down to 12 [30]. Of these principles the most important ones in relation to well-being at work are: 1) Build projects around motivated individuals, give them the environment and support they need and trust them to get the work done; 2) Promote sustainable pace: be able to maintain a constant pace indefinitely; 3) Best results emerge from self-organized teams; and 4) Teams reflect regularly when and how to improve. Self-organizing teams on one hand require and on the other produce motivated personnel.

The agile principles have been implemented through different practices. It is the way agile practices are applied that determines whether they are beneficial or harmful to well-being at work. Porchen [12] states that opportunities of agile methods have been in focus, but now, the view is shifting to risks.

In the Shine Technologies' survey [24], "people over processes" was the most liked feature of agile processes and "lack of authority" one of the least liked. According to Smith and Oltmann [25] the environment for flexibility requires putting the people and team first: the right people, commitment and dedication and adequate authority. Within Crystal people, interaction, community, skills, talents, and communication are considered as most important [27].

Also, for example XP states its own values (e.g., communication, feedback, courage and respect) and principles (e.g., embracing change). Courage may mean, e.g., the development team's courage to resist pressure to make unrealistic commitments [27]. Informal communication channels in agile methods include co-located teams, pair programming and daily stand-ups [7]. XP promotes teamwork by the fundamental that all software is produced in pairs, two programmers at one screen [3]. The goal is also not to force team members to specialize – every XP programmer participates in all of activities. Also, differences exist, e.g., Scrum prescribes cross-functional teams while Kanban allows specialist teams [12].

When applying agile methodology there are two main changing forces: continuous development of agile team's work processes and introducing agile to the organization. When considering the inception of agile methods it is important for managers to understand the factors that affect the adoption and its consequences to well-being at work. Also, in the operation phase it is important to know how to promote their use in a way that supports well-being at work.

B. *Well-being at Work when Using Agile*

Agile principles promote well-being in theory and it is commonly believed they increase the well-being of the developers. Agile methods hold many promises in relation to well-being at work, for example human centricity, interaction, and steady workload. However, they may also implicate strain, such as lack of recovery time and unfit management culture. Customer-drive, continuous reacting, changing goals, flexibility, culture change and new practices challenge well-being and expose to strain. In this paper, we understand well-being at work through three viewpoints defined by Gerlander and Launis [9]: avoiding excessive strain at work, feeling of autonomous and meaningful work, and change and development in work. Related ways of understanding well-being at work that have been studied within software engineering include motivation, job satisfaction, and employee retention, for example.

Avoiding excessive *strain* is one aspect of well-being at work. It means a balance between one's tasks and capabilities, work that matches the capabilities both qualitatively (not too challenging or too easy) and quantitatively (not too much or too little work) [11]. The balance theory evaluates the potential positive and negative impacts that could alter the balance of work system elements and result in stress load experienced by agile teams [28]. For example, when operating with XP practices, like 40-hour work weeks, it enables teams to work and maintain a sustainable pace [14]. Working overtime for a short time is accepted, but productivity collapses if teams work overtime for long periods. XP teams do not work excessive overtime for long periods of time [27].

Mann and Maurer's [15] results indicate that after the introduction of a Scrum process into an existing software development organization the amount of overtime decreased. This allows the developers to work at a more sustainable pace. Risks of agile methods still include self-intensification, overworking oneself and the threat to work-life balance [20], even though agile methods explicitly try to avoid them. There is evidence that balancing resources and workload (optimal resource allocation) is a labor-intensive and error-prone task [26]. Sherehiy [23] also found that a combination of job demands and job uncertainty have a significant effect on workforce agility. She suggests that a high level of uncertainty may increase perceived job demands and impede adaptivity at work.

The second core aspect of well-being at work is the *subjective experience of meaningfulness and autonomy of work* [cf. 2]. Within this aspect the focus is on the individual experiences and feelings of work, work practices and community. According to Mah and Lunt [12] creating quality with clean code means taking pride in what you do, without compromising one's professionalism. Sherehiy's [23] results revealed that the autonomy at work is one of the most important predictors of workforce agility, as well as well-being.

C. *Considering of Well-being at Work when Planning to Implement Agile*

Work organization is a main factor to anticipate meaningfulness and autonomy: governing practices (e.g., objectives, purposes, meanings), coordination procedures (e.g., work distribution methods, processes), and surveillance routines (e.g., monitoring rituals, standards) [6]. Sherehiy [23] suggests that if the management implements agile strategies in a way that positively affects job autonomy, job uncertainty, and employees' collaboration, it is more likely that employees will be able to perform a job in an adaptive and flexible way. Also, Maruping, Venkatesh and Agarwal [17] argue that the most effective control models are those that provide teams with autonomy in determining the methods for achieving project objectives.

It has also been shown that agile team could attain its flexible way of working only with the autonomy of the team. That bundles up agile way of working and well-being at work. For example, in a study of video game programming, agile project practices were found to be more empowering and flexible than other management methods emphasizing more management control [6]. In a related study, when shifting over to more centralized control of projects in a corporate R&D function of an IT company, engineers generally reacted to the attempted introduction of a new regime by increasingly presenting themselves as distinct from management [10]. Developers may sometimes view using of agile processes as an attempt to micromanage [6]. The risks of agile methods include ignoring self-determination, rigid organizational structures and possibilities of selection and control [20].

The third aspect of well-being at work considered is the *development* and *change* of work. Developing capabilities are essential in becoming agile [13]. Qualitative changes at work and in its environment occur faster and faster, non-stop and take place simultaneously. The challenges to well-being at work emerge as discontinuous work flow and unexpected interruptions [19].

Mayfield [18] found that in the transition to an agile development methodology there was an initial period of decision uncertainty and anxiety but that it was only temporary. Since agile adoption involves a significant process and organizational change, it is critical to success to focus initially on the human and cultural issues involved [21]. Boisnier and Chatman [5] propose that organizations may still benefit from simultaneously managing strong, stable cultures while maintaining the flexibility and adaptability necessary to survive the ebbs and flows of turbulent environments. When introducing agile methods, management practices and tools, motivated business experts, and common methods of managing change are needed in order to realize change and avoid the chaos caused by unpredictability and complexity [1].

Briand and Hodgson [6] identify agile methods as flexible, empowering and post-bureaucratic and non-hierarchical – as an attempt to mitigate the formal inflexibility of traditional project management to fit the demands of software creation. Agility literature emphasizes the importance of the development of a flexible, adaptable and highly knowledgeable workforce that is able to deal with unexpected and uncertain situations [23]. Teams operating within the context of agile are characterized as multifunctional, dynamic, and cooperative [28].

III. RESEARCH SETTING

In this section, a case research setting – planned for studying the connections between agile methods and well-being at work – is presented. The research targets are the experiences of the reality of agile practices and their perceived implications for well-being at work. The topic is analyzed through the following questions: 1) How do agile working practices in project management advance well-being at work? Do agile management principles support the well-being of agile team members? 2) What kind of challenges agile practices bring to maintaining well-being at work and sustainable productive work? The objective is to analyze the connection between agile work practices and well-being at work, and based on this analysis provide a model for adopting practices that enhance the well-being at work in product and service development.

Our preliminary hypotheses are, that when the agile practices are applied correctly: 1) they help to keep work strain steady during a working period (e.g., sprint), 2), they maintain and promote meaningfulness and autonomy of work and 3) they diminish discontinuity and interruptions at work and make development of work more fluent and natural part of work.

Factors of agile methods that produce and challenge well-being at work of the team are studied in three case companies. The research methods include a web based survey of well-being at work, physiological stress indicators and interviews of team members and supervisors. The outcomes of the research are the perceptions of applying agile and evidence based new knowledge with objective established methods.

With a web based survey agile methods of software development are explored through team members' experiences. Well-being is studied with established measures and taking advantage of existing well-being and agile surveys. By physiological stress measurements stress levels felt during the agile projects can be measured. In interviews of team members there are questions of applying agile practices, perceived well-being at work in general, experiences of well-being at work when applying agile practices, and expectations and needs to develop of agile practices.

IV. CONCLUSION AND FURTHER WORK

Through the methods described a holistic measure of well-being at work, applying agile methods, and managerial implications will be developed. The development of these issues takes use of a literature review, collection existing measures and results from the case study. In the future, the validation of the holistic measure also needs a wider statistical background from different kinds of agile teams.

V. ACKNOWLEDGEMENTS

This WIP paper is a part of Agile and Lean Product Development Methods for Embedded ICT Systems project (2012–2014), funded by TEKES – the Finnish Funding Agency for Technology and Innovation. Partners of the project are: BA Group, Ericsson, Finnish Institute of Occupational Health, FiSMA, Lindorff Finland, Neoxen Systems, Nextfour Group, Nordic ID and University of Turku.

REFERENCES

[1] P. Abrahamsson, Agile software development introduction: Introduction, current status & future. Jyväskylä, Finland: VTT Electronics, 2005.

[2] A. Antonovsky, Unraveling the mystery of health? How people manage stress and stay well. San Francisco: Jossey-Bass Publishers, 1987.

[3] K. Beck, Extreme programming explained. Boston: Addison-Wesley, 1999.

[4] M. S. Bird, "Utilizing agile software development as an effective and efficient process to reduce development time and maintain quality software delivery," Dissertation Abstracts International: Section B: The Sciences and Engineering, vol. 71, no. 5–B, 2010, 3329.

[5] A. Boisnier and J. A. Chatman, "The role of subcultures in agile organizations," in Leading and managing people in the dynamic organization, R. D. Day, R. S. Peterson, and E. A. Mannix, Eds. Mahwah, NJ, US: Lawrence Erlaum Associates Publishers, 2003, pp. 87–112.

[6] L. Briand and D.E Hodgson, "Management of creative projects or creative project management? Agile projects as micro-emancipation," Paper presented at the 7th international Critical Management Studies Conference, Naples, Italy, July 2011.

[7] A. Cockburn, "Agile software development joins the "would-be" crowd," Cutter IT Journal, vol. 15, no. 1, 2002, pp. 6–12.

[8] T. Dybå and T. Dingsøyr, "Empirical studies of agile software development: A systematic review," Information and Software Technology, vol. 50, no. 9–10, 2008, pp. 833–859.

[9] E.-M. Gerlander and K. Launis, "Työhyvinvoinnin tarkasteluikkunat," Työelämän tutkimus, vol. 5, no. 3, 2007, pp. 202–212.

[10] P. Gleadle, D. E. Hodgson, and S. Storey, "Project managing R&D engineers: Assent and recalcitrance," Paper presented at the 7th International Critical Management Studies Conference, Naples, Italy, July 2011.

[11] R. A. Karasek, "Job demand, job decision latitude, and mental strain: Implications for job redesign," Administrative Science Quarterly, vol. 24, 1979, pp. 285–309.

[12] H. Kniberg, Kanban and Scrum – Making the most of both, 2009 http://www.crisp.se/file-uploads/Kanban-vs-Scrum.pdf, 20.8.2013

[13] V. P Kochikar and M. P. Ravindra, "Developing the capability to be agile," Organization Development Journal, vol. 25, no. 4, 2007, pp. 127–134.

[14] M. Mah and M. Lunt, How agile projects measure up, and what this means to you, The Cutter Consortium Executive Report, 2009.

[15] C. Mann and M. Maurer, "A case study on the impact of Scrum on overtime and customer satisfaction," Proc. The Agile Development Conference (ADC '05), July 2005, pp. 70–79, doi: 10.1109/ADC.2005.1

[16] A. Marchenko and P. Abrahamsson, "Scrum in a multiproject environment: An ethnographically-inspired case study on the adoption challenges," Proc. AGILE '08 Conference, Aug. 2008, pp. 15–26, doi: 10.1109/Agile.2008.77

[17] L. M. Maruping, V. Venkatesh, and R. Agarwal, "A control theory perspective on agile methodology use and changing user requirements," Information Systems Research, vol. 20, no. 3, 2009, pp. 377–399.

[18] K. M. Mayfield, "Project managers' experience and description of decision uncertainty associated with the agile software development methodology: A phenomenological study," Dissertation Abstracts International: Section B: The Sciences and Engineering, vol. 71 (12–B), 2011, 7772.

[19] J. Mäkitalo, Work-related well-being in the transformation of nursing home work. Oulu, Finland: University of Oulu, D 837, 2005.

[20] S. Porschen, "Management of the informal by cooperative transfer of experience," in Innovation management by promoting the informal: Artistic, experience-based, playful, F. Böhle, M. Bürgermeister, and S. Porschen, Eds. Dordrecht: Springer, 2012, pp. 105–142.

[21] QSMA, The agile impact report. Proven performance metrics from the agile enterprise, A report from QSMA Associates, 2008.

[22] O. Salo and P. Abrahamsson, Agile methods in European embedded software development organisations: A survey on the actual use and usefulness of Extreme Programming and Scrum. Oulu, Finland: VTT Tech. Res. Centre of Finland, vol. 2, no. 1, 2006, pp. 58–64, doi: 10.1049/iet-sen:20070038

[23] B. Sherehiy, "Relationships between agility strategy, work organization and workforce agility," Dissertation, Kentucky, University of Louisville, 2008 http://louisville.edu/speed/industrial/academics/Bohdana%20Dissertation.pdf 20.8.2013

[24] Shine Technology's Agile Adoption Survey, 2003 http://www.shinetech.com/agile_survey_results.jsp 20.8.2013

[25] P. G. Smith and J. Oltmann, "Flexible project management: extending agile projects beyond software projects," Proc. PMI Global Congress, Oct. 2010, http://strategy2market.com/Preston-Smith/Articles/PMI/Flexible-Project-Management-Congress.pdf 20.8.2013

[26] Á. Szöke, "Decision Support for Iteration Scheduling in Agile Environments," Lecture Notes in Business Information Processing, vol. 32, 2009, pp. 156–170.

[27] L. Williams, A Survey of Agile Development Methodologies, 2007 http://agile.csc.ncsu.edu/SEMaterials/AgileMethods.pdf 20.8.2013

[28] C. A. Yauch, "Team-based work and work system balance in the context of agile manufacturing," Applied Ergonomics, vol. 38, no. 1, 2007, pp. 19–27.

[29] www.agilemanifesto.org Manifesto for Agile Software Development, 20.8.2013.

[30] www.agilemanifesto.org/principles.html Principles behind the Agile Manifesto, 20.8.2013.

The Scrum Product Owner – Customer Collaboration & Prioritizing Requirements

Empirical research in a sample of Irish Industry

Trish O'Connell
Department of Information Technology
National University of Ireland, Galway, Ireland.
trish.oconnell@nuigalway.ie

***Abstract*— The existing body of literature on Agile Scrum is extensive. Many authors, ([1], [2], [3], [4]), concur that the role of the Product Owner is to represent the customers' requirements to the development team and set the priorities for the work to be completed. The Agile Manifesto specifies customer collaboration as being of more importance than contract negotiation. So, we might expect that in addition to setting priorities the Product Owner role in Scrum would work closely with the Customer. This paper investigates a sample of Irish software development organizations to determine the level of adherence to Agile Scrum guidelines with regard to the two key aspects of customer collaboration and requirements prioritization.**

Keywords-Agile; Scrum; Requirements Product Owner; Customer collaboration;

I. Introduction

In the absence of any *a priori* knowledge, it is generally believed that if companies claim to be Agile then they are, in fact, following the precepts and guidelines of their chosen Agile methodology, whether this be Scrum [5], eXtreme Programming [6], Crystal Clear [7] or indeed any of a plethora of Agile practices.

It may then come as something of a surprise to discover that "there is often a substantial difference between the textbook 'vanilla' version" [8] of the method and the method actually enacted in practice, what Senapathi et al [9] refer to as the "method-in-action." Conboy & Duarte [8] elaborate: "Prescribed practices are inevitably interspersed in diverse ways or tailored to suit the specific needs of teams." So, can self-described Agile enterprises really lay claim to being Agile or are they, perhaps, using an *ad hoc* approach which pays lip service to Agile principles with the (unintentional) benefit of keeping the stakeholders happily deluded? To what extent do companies that describe themselves as being Agile actually follow Agile guidelines as documented by the pioneers of the various Agile methodologies?

Many authors [38], [39] agree that lack of user involvement is a primary cause of project failure. The CHAOS report of 2010 [40] stated: "projects that lack user involvement perform poorly." Consequently, the degree of user involvement in organizations that describe themselves as being Agile was of immediate interest to the author.

This paper, based on empirical research, examines the author's contention that because "agile methodologies intentionally leave a lot to be defined about exactly how the methodology is implemented" [4], what results is sometimes an extemporized approach to implementing Agile methods with a resultant lack of project success. Because many of the Agile practices are somewhat loosely, if at all, defined, it is possible that some organizations might take this as carte blanche to omit some of the fundamental aspects that made Agile so pertinent at the outset.

The research hypothesis of this work was to ascertain if this ad hoc implementation of Agile methods extended to the customer involvement / Product Owner domain.

Whilst it may be argued that a plan based or prescribed method of developing software might not always be easy to work with due to constantly changing customer requirements or "software requirements churn" [10], Addison & Vallabh [11] advocate that to control software projects it is important to "develop and adhere to a software development plan." Fitzgerald [12] also contributes to this argument citing that "experienced developers are more likely to use a methodology, as they would be aware of its benefits". Fitzgerald [12] further claims that "inexperienced developers are more likely to follow a methodology rigorously", perhaps because it lends structure to an otherwise chaotic endeavour.

Thus, although it is widely accepted that "standard software development models often provide explicit detailed guidelines" [13], the author decided to conduct some quantitative research into aspects of actual Agile implementation in a sample of Irish software industry with a view to gaining an understanding of the level of compliance to documented Agile precepts. In the interests of brevity, this paper will deal only with the Scrum Product Owner, prioritization of customer requirements and customer collaboration aspects.

Section II of this paper briefly outlines the background to one of the foremost Agile methods, Scrum, which incorporates the role of Product Owner. Section III briefly describes the research design of the study. Section IV presents the results of the study and this is followed by a discussion of the findings in Section V.

II. Background

Agile software development methods emerged in the late 1990s with the Agile Manifesto [14] being published in 2001 (http://agilemanifesto.org/). There are many different approaches to implementing Agile and each has its own 'vanilla' version. Sutherland [15] explains "Each Agile methodology has a slightly different approach for implementing the core values from the Agile Manifesto, just as many computer languages manifest the core features of object-oriented programming in different ways." The methodologies chosen for the study were Scrum and XP, since previous work in this domain by Bustard [16] identified these as the most prominent of the Agile methodologies currently in use. Salo, & Abrahamsson [17] refer to Scrum and XP as the "perhaps best known agile methods". However, in the interest of brevity only Scrum will be discussed in this paper.

A. *Scrum*

According to Ken Schwaber [5] (co-creator of Scrum with Jeff Sutherland), "Scrum is an enhancement of the commonly used iterative/incremental object-oriented development cycle." It is more of a framework than a methodology but it nevertheless takes, according to Millett et al [18], an "iterative approach to software development." Sutherland [15] explains Scrum "structures development in cycles of work called Sprints. These iterations are no more than one month each, and take place one after the other without pause. The Sprints are timeboxed – they end on a specific date whether the work has been completed or not, and are never extended".

Schwaber [19] describes product requirements as being "contained in an ordered list known as the Product Backlog." At the beginning of each Sprint, the requirements are prioritized into a list known as the Sprint Backlog with the aim of completing an agreed set of deliverables by the end of the Sprint. Sutherland [15] explains further, "During the Sprint, the chosen items do not change. Every day the team gathers briefly to inspect its progress, and adjust the next steps needed to complete the work remaining. At the end of the Sprint, the team reviews the Sprint with stakeholders, and demonstrates what it has built. People obtain feedback that can be incorporated in the next Sprint. Scrum emphasizes working product at the end of the Sprint that is really "done"; in the case of software, this means code that is integrated, fully tested and potentially shippable."

Barari [20] advises that "it is important to follow the guidelines defined in Scrum but the ultimate goal is to deliver what you promised". With regard to the guidelines, Schatz & Abdelschafi [21] state quite categorically that "there aren't many rules in Scrum but you need to adhere to the ones that (do) exist". Unfortunately, the rules of transitioning software development from a plan-driven approach to an Agile approach are not set in stone and this may be where the confusion lies. The 'rules' that exist are the implementation of the 12 principles set out in the Agile Manifesto [14]. It is the author's opinion that it is the interpretation of these rules that is often confusing and sometimes even problematic.

Most authors on Agile ([1], [3], [22], [23]) agree that the Scrum framework should include a Product Owner. The role of the Product Owner will now be reviewed.

B. *THE PRODUCT OWNER*

According to Deemer et al [24] "The Product Owner is responsible for maximizing return on investment (ROI) by identifying product features, translating these into a prioritized list, deciding which should be at the top of the list for the next Sprint, and continually re-prioritizing and refining the list. The Product Owner has profit and loss responsibility for the product, assuming it is a commercial product. In the case of an internal application, the Product Owner is not responsible for ROI in the sense of a commercial product (that will generate revenue), but they are still responsible for maximizing ROI in the sense of choosing – in each Sprint – the highest-business-value lowest-cost items". How well this focus on the "highest-business-value lowest-cost items" correlates with the customers' requirements is, in this author's opinion, debatable. Deemer et al [24] offer the opinion that "'value' is a fuzzy term and prioritization may be influenced by the desire to satisfy key customers." Thus, the role of the Product Owner in Scrum might not appear to be as clear cut as the original proponents of Agile might have wished.

Stober & Hansmann [3] concur and define a Product Owner who "represents the stakeholders, such as customers." Consequently, it might be apposite to assume that there should be a tenable link between the Product Owner and the customer.

However, Sutherland [25] identifies a ubiquitous dilemma... "In some cases, the Product Owner and the customer is the same person; this is common for internal applications. In others, the customer might be millions of people with a variety of needs, in which case the Product Owner role is similar to the Product Manager or Product Marketing Manager position in many product organizations. However, the Product Owner is somewhat different than a traditional Product Manager because they actively and frequently interact with the Team, personally offering the priorities and reviewing the results of each two- or four-week iteration, rather than delegating development decisions to a Project Manager". Deemer et al [24] summarize, "It is important to note that in Scrum there is one and only one person who serves as – and has the final authority of – Product Owner, and he or she is responsible for the value of the work". Schwaber [26] describes the Product Owner as "the single wringable neck". Insofar as it is the Product Owner who represents the customer requirements to the development team, the success or failure of the project can ultimately be attributed to this one individual. Beyer [2] sees the Product Owner as "the customer representative" and outlines his responsibility to "find out what the stakeholders and end users actually need." Having requirements which are "prioritized by the product owner" [3] is yet another prerequisite of Scrum. In the Scrum approach, according to Cohn [27], "requirements are maintained in a backlog, called the Product Backlog, prioritized by business value." Having been prioritized, the work (or as much of it as possible) is accomplished by the Scrum development team in fixed timeframes "known as Sprints" [27], which usually last two to four weeks, depending on the product or service. Items are taken off the backlog in priority order to be worked on as parts of the Sprint Backlog in the current iteration. At the end of the Sprint, there is usually a Sprint review [22], where the team demonstrates what it has accomplished to the customer with a view to soliciting feedback.

According to Schwaber [19], the Product Owner is "responsible for representing the interests of everyone with a stake in the project and its resulting system." Many of the proponents of Scrum, including [4], advocate "as much customer collaboration as possible" but he counsels that the "Product Owner represents the voice of the customer and is expected to provide overall direction to guide the project toward producing the value to satisfy customer needs" [4]. This should most likely involve close collaboration with customers and stakeholders. In most Scrum training workshops, it is advised to ensure customer involvement throughout the development process. This is often referred to as capturing the "voice of the customer" [28] in an attempt to deliver the required content. It has been widely accepted [29] that customer involvement is critical to successful software development. In fact, Paetsch, Eberlein et al [30] state "customer involvement was found to be the number one reason for project success, while the lack of user involvement was the main reason given for projects that ran into difficulties."

III. RESEARCH METHOD

The research on which this paper is based was conducted as a quantitative study that was descriptive in nature. Leedy & Ormrod [31] describe this type of research as "identifying the characteristics or exploring possible correlations among two or more phenomena." The authors also state that "descriptive research examines a situation as it is." However, as Oppenheim [32] explains, "no valid causal interpretations are possible", thus, whilst the data collected may describe the actual situation, the research is limited to being solely a descriptive analysis.

There are many ways to conduct descriptive quantitative research. Thomas [37] refers to three methods: surveys, correlation analysis and experiments whilst Leedy & Ormrod [31] also include "observational studies and developmental designs". Having reviewed the suitability of each of these methods it was decided to use an online survey to collect primary research data. Leedy & Ormrod [31] explain that a survey "involves acquiring information about one or more groups of people by asking them questions and tabulating their answers". The authors indicate that "the ultimate goal is to learn about a large population by surveying a sample of that population." It needs be stressed, however, that survey research "captures a fleeting moment in time" [33]. It is possible that the response to a particular question might be totally different in two or three months' time. Once this precept was understood, however, it was felt that a survey would be a perfectly acceptable way to discover information about the topic to be investigated. De Vaus [34] states, "Survey research is widely regarded as being inherently quantitative and positivistic and is contrasted to qualitative methods that involve participant observation, unstructured interviewing, case studies, focus groups, etc. Quantitative survey research is sometimes portrayed as being sterile and unimaginative but well suited to providing certain types of factual, descriptive information – the hard evidence."

If survey research has a drawback it would seem to be that the results are dependent on the participants' willingness to participate, in addition to their ability to correctly answer the questions asked. Leedy & Ormrod [31] refer to the fact that the method relies on "self report" data. The authors caution that "people are telling us what they believe to be true or, perhaps, what they think we want to hear." Survey research can be conducted via a number of different methods: the face-to-face interview, the telephone interview or the documented questionnaire, which can be either paper or Internet based. As Salo & Abrahamsson [17] note "web-based data collection also overcomes some limitations of ordinary mail surveys and other data collection mechanisms in terms of speed and cost." It was also planned that a limited amount of interviewing would be required to ensure that the correct conclusions were drawn. Thus, to conduct research into this domain a sample of software professionals at both management and Scrum team level in a cross section of Irish Software development companies, who profess to use Scrum, were polled for their perspectives. This is described next.

A. *THE PARTICIPANTS*

In an ideal scenario, it would be preferable to obtain a totally random selection of Irish software development companies to answer the research questions. However, given the likelihood that the response rate would be low (which is one of the main drawbacks of this research method, what Leedy & Ormrod [31] refer to as "low return rate"), it was decided to indulge in a degree of "purposive sampling" [35]. Nardi [35] explains purposive sampling as sampling one or more specific pre-defined groups. This approach was adopted as it was felt to be important to collect data on organizations that had some prior knowledge of Agile practices as opposed to taking a completely random sample, which may have resulted in confused responses. To generate survey data, a random sample of software companies was targeted from software groups known to be somewhat familiar with the concepts of Agile software development, groups such as AgileIreland, Information Technology Association Galway (ITAG), the Irish Software Association (ISA), the Irish Software Innovation Network (ISIN) training companies, blogs etc. All of these were contacted to host the online survey on their websites, where it would be visible to their members.

Using these organizations, it was possible to distribute the online survey to a diverse audience of software development professionals who had an established history with, or at the very least, a passing knowledge of, Agile and who, it was hoped, would be more likely to respond to the questions. In an attempt to capture a representative view, cross-functional participants, including both Scrum team members and software development management in organizations that use Scrum, were targeted. In this way it was hoped that the findings would be representative of the actual state of play of software development in Irish industry. The breakdown of Scrum team participants is shown in Table I.

TABLE I. SCRUM TEAM SURVEY PARTICIPANTS

	Organization Size		
Role	***1 to 50***	***51 to 500***	***500+***
Designer	1	1	1
Senior Developer	2	3	1
Developer	2	4	3
Test Engineer	2	3	2

Similarly, the breakdown of Scrum management participants is shown in Table II.

TABLE II. MANAGEMENT SURVEY PARTICIPANTS

	Organization Size		
Role	***1 to 50***	***51 to 500***	***500+***
S/W Dev. Mgr.	3	3	4
Project Mgr.	2	2	4
Q.A. Mgr.			1
Test Mgr.		1	

Given the fact that the survey was online, it was not possible to compute a response rate, *per se*. However, it was felt that a sufficiently representative number of respondents had contributed to make the results relevant.

IV. RESULTS

Whilst all Scrum teams admitted having a Product Owner it became clear that the Scrum teams were not always aware of the link between the Customer and the Product Owner. When asked how frequently the Product Owners consulted with the customer the responses were as given in Table III.

TABLE III. LEVEL OF AWARENESS OF INVOLVEMENT BETWEEN PRODUCT OWNER AND CUSTOMER

Unaware of involvement	44%
Aware of weekly involvement	20%
Aware of infrequent involvement	8%
As required	28%

Although, in theory, the Product Owner sets the vision for the product and is responsible for prioritizing requirements for the team to work on for the Sprint duration, in practise it was found that for 44% of those who described themselves as being Scrum team members this did not happen. In fact, it transpired that in some cases requirements were prioritized as shown in Table IV.

TABLE IV. PRIORITIZATION OF REQUIREMENTS

Product Owner	56%
Scrum Master	24%
Release Manager	12%
Combination	8%

One interesting comment was that the developer didn't know how priorities were set, but felt that there was a "mysterious process in operation."

When questioned about the involvement of customers at Sprint reviews the Scrum teams' responses were as shown in Table V.

TABLE V. CUSTOMER INVOLVEMENT AT SPRINT REVIEWS

No customers in attendance	12%
Unsure of attendance	4%
Customers in attendance	84%

From the perspective of the developers with regard to customer involvement, it would appear that 16% felt the involvement of customers was either not encouraged, or they were unaware of any efforts to involve customers.

When management at self-described Agile organizations were asked if customer involvement was encouraged (in the form of attendance at Sprints etc) 13% admitted that this was not the case.

V. DISCUSSION

The research effectively offers a snapshot of Irish software industry over the duration of the survey availability window, which was two months from July to August, 2011. Although the sample was not as large as had been envisaged, and it is consequently not possible to make generalizations from the results, it is nonetheless valid to make some observations. When taken in isolation, the Scrum results presented in section IV are somewhat disconcerting; however, they are largely in line with what was expected. It should be noted that the results are not skewed by the presence of a number of responses from organizations who are not using any of the Scrum precepts. A correlation of all of the responses would seem to show that only 12% of those who responded were operating precisely to the Scrum guidelines. The remainder had, indeed, adopted an *ad hoc* approach to Scrum for whatever reason. This might, in part, be the reason behind failed Agile projects.

For any Agile method the theory would seem to indicate that user involvement is crucial. In fact, one might go further than mere user involvement, and in order to gain valuable feedback to the project, cite user participation as being key to a successful software development initiative. Kautz [36] acknowledges "Agile development practices and principles insist on the customer taking control and being constantly involved." This is underpinned by the Agile Manifesto [14], which advocates "Customer collaboration over contract negotiation."

Paetsch, Eberlein et al [30] concur, "All agile approaches emphasize that talking to the customer is the best way to get information needed for development and to avoid misunderstandings. If anything is not clear or only vaguely defined, team members should talk to the responsible person and avoid chains of knowledge transfer. Direct interaction also helps establishing trust relationships between customers and developers."

However, in the vast majority of companies it is thought not to be feasible to have the customer on site or actively involved as described. The solution to this in most companies is to appoint a customer proxy. However, Beyer [3] says "Product Owners as defined by Scrum do not make good user surrogates. They may be responsible for representing all the stakeholders of a system, including end-users, the customer who makes the purchase decision, and the internal stakeholders. But they are not any of these people."

The findings of the study answered the primary research question and found that, as expected, the adoption of Scrum by many organizations was not as rigorous as the proponents of the methodology might have wished. The implications of this approach to software development could have many ramifications not least being poor Scrum team morale, projects being late and/or not delivering what the customer requires.

In the author's opinion, the results of the survey highlight the need for further research. In particular, it is important to acknowledge that the results of this study were based on a relatively small sample of Irish software industry due largely to the aggressive timeframe in which the author operated. Whilst the preliminary research commenced in February 2011 the completion deadline for the thesis was in August of the same year. It would, indeed, be interesting to investigate whether the findings would be replicated on a larger set of software development organisations.

ACKNOWLEDGMENT

The author would like to thank Ms. Karen Young (Discipline of IT, National University of Galway, Ireland) for her unflagging patience in keeping this research going for the duration of the thesis portion of the MScSED.

REFERENCES

[1] J. Highsmith, Adaptive Software Development Ecosystems. Boston, MA: Pearson Education Inc, 2002, pp. 244-245.

[2] H. Beyer, User Centered Agile Methods, Synthesis Lectures on Human-Centered Informatics. Ed. J. Carroll, Morgan & Claypool, 2010, p.4.

[3] T. Stober and U. Hansmann, Agile Software Development: Best Practices for Large Software Development Projects. Berlin Heidelberg: Springer-Verlag, 2010, p. 41.

[4] C.G. Cobb, Making Sense of Agile Project Management Balancing Control and Agility. Hoboken, NY: Wiley & Sons, 2011, pp. 101-103.

[5] K. Schwaber, "Scrum Development Process", Proc. OOPSLA'95 Workshop on Business Object Design and Implementation, Austin, Texas, USA, 1995, pp. 117-134.

[6] K. Beck, Extreme Programming Explained: Embrace Change, Boston, MA: Addison Wesley, 1999.

[7] A. Cockburn, Agile Software Development, Boston, MA: Addison-Wesley, 2001.

[8] K. Conboy and V. Duarte, "Scaling Agile to Lean – Track Summary", Proc. LESS 2010, Helsinki, Finland, Volume 65, 2010, pp 1-2.

[9] M. Senapathi, P. Middleton, and G. Evans, "Factors Affecting Effectiveness of Agile Usage – Insights from the BBC Worldwide Case Study", Proc.12th International Conference, XP 2011, Madrid, Spain, 2011, p. 136.

[10] J. Dooley, Software Development and Professional Practice. New York, NY: Apress. 2011.

[11] T. Addison and S. Valabh, "Controlling Software Project Risks – an Empirical Study of Methods used by Experienced Project Managers", Proc. SAICSIT 2002, Port Elizabeth, South Africa, pp. 128-140.

[12] B. Fitzgerald, "The use of systems development methodologies in practice". Info Sys Journal (1997) Vol 7, pp. 201-212.

[13] J.D. Fontana, "Models of Software Evolution Life Cycle and Process", Technical Document 1893, July 1990. Retrieved Aug. 16th, 2013 from

http://www.dtic.mil/dtic/tr/fulltext/u2/a227328.pdf.

[14] M. Fowler and J. Highsmith, "The Agile Manifesto", Software Development, Vol. 9 No. 8 (August, 2001) pp. 28-32.

[15] J. Sutherland, "The Scrum Handbook", Somerville MA: Scrum Training Institute, 2010.

[16] D. Bustard, "Beyond Mainstream Adoption: From Agile Software Development to Agile Organizational Change", Proc. IEEE 19th International Conference ECBS, Serbia 2012, pp. 90-97.

[17] O. Salo and P. Abrahamsson, "Agile methods in European embedded software development organisations: a survey on the actual use and usefulness of Extreme Programming and Scrum", Proc. IET Software, 2008, Vol. 2, Issue 1, pp. 58–64, doi: 10.1049/iet-sen:20070038.

[18] S. Millett, J. Blankenship, and M. Bussa, Pro Agile. NET Development with SCRUM. New York, NY: Apress, 2011, p.13.

[19] K. Schwaber, Agile Project Management with Scrum. Redmond, Washington: Microsoft Press, 2009, p10.

[20] T. Barari, "Tips for First Time Scrum Masters" Scrum Alliance, 2009. Retrieved 16th Aug. 2013 from

http://www.scrumalliance.org/community/articles/2009/may/tips-for-first-time-scrummasters.

[21] B. Schatz and I. Abdelschafi, "Primavera Gets Agile: A Successful Transition to Agile Development", IEEE Software. 2005, May/June Vol. 22 no. 3, pp36-42,

doi:10.1109/MS.2005.74.

[22] R. Pichler, Agile Product Management with Scrum: Creating Products that Customers Love. Boston, MA: Pearson Education Inc, 2010, pp. 2-4.

[23] K. Schwaber and M. Beedle, Agile Software Development with Scrum. NJ: Prentice Hall, 2001, pp. 34-35.

[24] P. Deemer, G. Benefield, C. Larman, and B. Vodde, "Scrum Primer v 1.2" Retrieved 17th Aug, 2013 from

http://goodagile.com/scrumprimer/scrumprimer.pdf.

[25] J. Sutherland, The Scrum Papers: Nuts, Bolts and Origins of an Agile Framework. Cambridge, MA: Scrum, Inc., 2012, p.15.

[26] K. Schwaber, The Enterprise and Scrum. Redmond, Washington: Microsoft Press, 2011, p6.

[27] M. Cohn, Succeeding with Agile : Software Development using Scrum. Upper Saddle River, New Jersey: Addison-Wesley. pp. 257–284. 2010.

[28] J.R. Hauser and D. Clausing, "The House of Quality", Harvard Business Review (May–June), 1988, pp. 63–73.

[29] Standish Group International, Chaos Report 2000. Retrieved Aug 18th, 2013, from http://www.gobookee.net/standish-group-chaos-report-2000/.

[30] F. Paetsch, A. Eberlein, and F. Maurer, "Requirements Engineering and Agile Software Development", Proc. of the 12th IEEE international Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, June 2003, pp. 308 – 313.

[31] P.D. Leedy and J.E. Ormrod, Practical Research Planning and Design. New Jersey: Prentice Hall, 2005 p.179.

[32] A.N. Oppenheim, Questionnaire Design, Interviewing and Attitude Measurement, New York: Continuum, 2005, p.4.

[33] C.A. Mertler, Action Research: Improving Schools and Empowering Educators. California: Sage Publications, California, 2006, p.95.

[34] D. DeVaus, Surveys in Social Research. New South Wales, Australia :Routledge, 2002, p. 5.

[35] P.M. Nardi, Doing Survey Research: A Guide to Quantitative Methods. Boston: Pearson Education 2003, p. 119.

[36] K. Kautz "Inclusive Design in Practice: A Study of Participatory Design, Customer and User Involvement in Agile Software

Development," Proc. of the 32nd Information Systems Research Seminar, Norway, August 2009, p217.

[37] R.M. Thomas, "Blending Qualitative and Quantitative Research Methods in Theses and Dissertations" Corwin Press, California. 2003.

[38] K.L. James, "Software Engineering", Delhi: PHI, 2008, p. 34.

[39] R.K. Wysocki, "Effective Management: Traditional, Agile, Extreme", Indianapolis, IN: Wiley& Sons, 2011, Ch. 16.

[40] Standish Group International, CHAOS Summary for 2010 retrieved Aug 17th, 2013, from http://insyght.com.au/special/2010CHAOSSummary.pdf.

Benefits and Limitations of Using the MPS.BR Model with Agile Methodologies

A Survey Based on a Systematic Literature Review

Robson Amorim de Souza
Department of Computer Science
Mato Grosso State University, UNEMAT
Barra do Bugres, Brazil
robson.unemat@gmail.com

Fernando Selleri Silva
Dept. of Computer Science, Mato Grosso State University
Center of Informatics, Federal University of Pernambuco
Barra do Bugres, Recife, Brazil
fss4@cin.ufpe.br

Felipe Santana Furtado Soares and Silvio Romero de Lemos Meira
Center of Informatics, Federal University of Pernambuco, UFPE
Recife Center of Advanced Studies and Systems, C.E.S.A.R
Recife, Brazil
fsfs@cin.ufpe.br, srlm@cin.ufpe.br

***Abstract*— This paper investigates the benefits and limitations of using the Reference Model for the Improvement of the Brazilian Software Process (MPS.BR) with agile methodologies. A survey of the Brazilian and international literature was performed, which used the concepts of a systematic literature review. Altogether 21 studies were included on the subject, viz., 12 articles, 1 Master's dissertation, 3 dissertation from post-graduate courses, 4 end-of-course undergraduate monographs, and 1 report arising from an undergraduate traineeship. Based on the results presented in the studies, agile methodologies and their practices were found to be feasible used in serving the initial levels of MPS.BR, but for the highest levels of the model, additional practices must be used.**

Keywords-MPS.BR; Brazilian SPI Model; Agile Methodologies; Suitability.

I. INTRODUCTION

Ensuring that their products or services are of good quality is essential if organizations have to survival on the market. Generally, this quality is related to the production processes of the product/service. In the context of Information Technology, all software must be of good quality, both in the development process and the product itself. In order to establish Software Process Improvement (SPI), various software development organizations are looking for quality reference models available on the market, such as: ISO/IEC 90003 - Guidelines for the application of ISO 9001:2000 to computer software [1], ISO/IEC 12207 - Software Life Cycle Processes [2], CMMI - Capability Maturity Model Integration [3], and MPS.BR - Brazilian Software Process Improvement [4]. However, quality models, generally, establish firstly "what" needs to be done in order to engage on demanding processes and secondly, methodologies for developing software that indicate "how" to do so.

Agile methodologies for use in software development became widely known from 2001, when a group of professionals, from the software area, assembled and published the Manifesto for Agile Software Development, also known as the Agile Manifesto [5]. These methodologies aim to develop software with high quality, iteratively and incrementally, thereby stimulating team interaction, with less documentation, and aim at meeting deadlines, costs and quality standards. Among various agile methodologies, the most used are Scrum and Extreme Programming (XP) [6].

In this context, this paper sets out to discuss by means of a survey of the literature that uses the concepts of a Systematic Literature Review, the Reference Model (RM) for Improving the Brazilian Software Process (MPS.BR) together with agile methodologies for software development. The following research question was considered: *What is known about the benefits and limitations of adopting the MPS.BR reference model using agile methodologies*? Further, the paper seeks to characterize academic production on the Brazilian model of quality assurance, together with agile methodologies.

The paper is organized in five sections. Section 2 gives a brief theoretical description of the MPS.BR model and agile methodologies. Section 3 describes the methodology used. Section 4 reports the results, and comments on benefits and limitations. Conclusions are drawn and recommendations for future work are made in Section 5.

II. THEORETICAL BACKGROUND

Initially, the MPS.BR model will be briefly described; then, the main concepts regarding agile methodologies are presented.

A. MPS.BR

MPS.BR is a Brazilian Software Process Improvement Program that was created in December 2003, by the

Association for Promoting Excellence in Brazilian Software (SOFTEX), with the support of several public and private organizations in Brazil, including: the Ministry of Science and Technology (MCT), an Agency that Funds Studies and Projects (FINEP), and the Inter-American Development Bank (IDB) [4]. MPS.BR aims to assist organizations, particularly small and medium-sized Brazilian companies, to achieve good quality in software development, in a smoother and less expensive way.

The MPS.BR program proposes the SPI Reference Model for Software (MR-MPS-SW), which defines seven maturity levels for the software process of an organization [4]: In descending order, these are A (In Optimization), B (Quantitatively Managed), C (Defined), D (Largely Defined), E (Partially Defined), F (Managed), and G (Partially Managed). For each maturity level, a profile of processes is assigned that suggests where the organization must make efforts to improve, as described below (for which the acronym is given in Portuguese, in brackets): In ascending order of maturity level, these are:

- Level G: Project Management (GPR) and Requirements Management (GRE).
- Level F: Acquisition (AQU), Configuration Management (GCO), Quality Assurance (GQA), Project Portfolio Management (GPP), and Measurement (MED).
- Level E: Evaluating and Improving the Organizational Process (AMP), Defining the Organizational Process (DFP), Human Resources Management (GRH), and Reuse Management (GRU).
- Level D: Requirements Development (DRE), Product Integration (ITP), Product Design and Construction (PCP), Validation (VAL), and Verification (VER).
- Level C: Development for Reuse (DRU), Management Decisions (GDE), and Risk Management (GRI).
- Level B: Project Management (GPR – evolution).
- Level A: (process optimization).

B. *Agile Methodologies*

Agile methodologies refer to approaches of software development used by organizations that focus on flexible collaboration, as they deal with projects in which requirements change constantly. Their core values were defined in the Agile Manifesto [5], as: individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation, and responding to change over following a plan.

Among the main agile methodologies, especially in Brazil, Scrum and Extreme Programming (XP) are foremost [7], the focus on each is as follows:

- Scrum: its focus is on managing software development, through an iterative and incremental process. It aims to deliver software in the shortest time, to meet deadlines and to reduce costs [8].
- Extreme Programming (XP) focuses on the development of a specific piece of software, by providing a set of practices that addresses the different phases of the life cycle, in an incremental and iterative format [9].

III. REVIEW METHOD

A survey of the literature on the MPS.BR model and agile methodologies cited in Brazilian and international sources was conducted, using the concepts of a Systematic Literature Review, described in Kitchenham and Charters [10]. This is a way to identify, evaluate and interpret the relevant papers available on a research question in particular, a topic area or phenomenon of interest. The systematic review process generally consists of identifying a research study by using a protocol (described in this section), study selection, quality assessment, data extraction and synthesis.

In this article, we used the stage of study selection, which includes (automatic and manual) search and the application of inclusion/exclusion criteria, as described below. Due to the limit on the available resources, the stage of quality assessment was suppressed. Data extraction and synthesis stages were performed, the findings of which revealed benefits and limitations.

We chose to search, in addition to articles published in journals and conferences, academic studies (works from undergraduate and post-graduate courses, Master's dissertations and PhD thesis). Although in the systematic review process, inclusions such as these are not common, mainly due to the review process being less formal, academic studies were considered because this enabled ongoing research in the area to be mapped.

A. *Search for studies*

The first activity for the search was to formulate a string, which makes an automatic search feasible. This string was set taking into account the research question addressed in Section 1, from which were derived the key terms, their synonyms or related words, as shown in Table 1.

TABLE I. TERMS USED IN SEARCHES

Term	Synonyms or related words
"MPS.BR"	"MPSBR", "MPS BR", "MPS-BR", and "Brazilian software process improvement"
"process"	"method" and "methodology"
"agile"	"agility", "light", "scrum", "extreme programming", "XP", "dynamic system development", "DSDM", "crystal", "kanban", "feature driven development", "FDD", "lean", "adaptive software development", "ASD", "test driven development", and "TDD"

The terms and their synonyms or related words were organized in a standard search string, in which each key term was grouped with the logical operator "AND" and its synonyms or related words with the operator "OR", as follows:

("MPS.BR" OR "MPSBR" OR "MPS BR" OR "MPS-BR" OR "Brazilian software process improvement") AND ("process" OR "method" OR "methodology") AND ("agile" OR "agility" OR "light" OR "scrum" OR "extreme

programming" OR "XP" OR "dynamic system development" OR "DSDM" OR "crystal" OR "kanban" OR "feature driven development" OR "FDD" OR "lean" OR "adaptive software development" OR "ASD" OR "test driven development" OR "TDD")

The next step was to define in which electronic databases to conduct searches and to include digital libraries of organizations that have an interest in the subject, search engines that index academic studies in Brazil and international mechanisms for indexing scientific studies. Some terms of the string were translated as per the language of the database language (Portuguese or English) in order to get better results. In some bases (national and international), the default string had to be adapted. However, the original essence of the string, without restricting the results, was preserved. The following databases were considered:

- Organizations: Association for Promoting Excellence in Brazilian Software (SOFTEX), Ministry of Science and Technology (MCT), and Brazilian Computer Society (SBC).
- National mechanisms: Dedalus – USP, Public Domain, Google Web Brazil, Google Scholar Brazil, and Scientific Electronic Library Online (SciELO).
- International mechanisms: ACM, Compendex, IEEE, ISI, Science Direct, Scopus, Springer, and Wiley.

The inclusion of the Google search engine is not common in most systematic reviews. However, it was included with the intention of facilitating the identification of academic studies originating from a wide variety of Higher Education institutions. The search in Google Web returned 980,000 results, of which only the first 200 results were considered, because from that point on, the results proved to be irrelevant and/or repetitive. In the other electronic databases, including Google Scholar, all returned results were considered.

The automatic search was conducted from April 28 to May 21, 2012, and included studies made available up to (and including) December 31, 2011. A summary of the results obtained is listed in Table 2, grouped by electronic database, and amounted to 836 in total.

TABLE II. AUTOMATIC SEARCH RESULTS

Eletronic Database	**Result**
MCT (www.mct.gov.br)	2
BDBComp – SBC (www.lbd.dcc.ufmg.br/bdbcomp/)	56
SOFTEX (www.softex.br)	58
Dedalus - USP (www.dedalus.usp.br)	177
Public Domain (www.dominiopublico.gov.br)	27
Google Web (www.google.com)	200
Google Scholar (scholar.google.com)	172
Scielo (www.scielo.org)	55
ACM (portal.acm.org/dl.cfm)	6
Compendex (www.engineeringvillage2.org)	11
IEEE (ieeexplore.ieee.org)	18
ISI (apps.isiknowledge.com)	3
Science Direct (www.sciencedirect.com)	4
Scopus (www.scopus.com/home.url)	17
Springer (www.springerlink.com)	29
Wiley (onlinelibrary.wiley.com)	1
Total	**836**

The survey also included a manual search, which was undertaken immediately after the automatic search, in the proceedings of the Brazilian Symposium on Software Quality (SBQS) and the Brazilian Symposium on Software Engineering (SBES). The manual search identified one potentially relevant study, published in SBQS 2009. Altogether 837 results were considered for being selected for study.

B. *Study Selection*

First of all, the titles and abstracts of the studies were analyzed in order to identify potentially relevant studies. After eliminating redundancies (studies returned by more than one database engine) and studies clearly irrelevant to this research, 56 studies were considered potentially relevant. The rationale for this reduction (837 results to 56 potentially relevant studies) was due to the redundancy of results arising from using two or more database engines and due to the extensive coverage of the string, which returned studies with the terms, e.g., "MPS.BR", "process" and "agile", applied in a different context to that of the objective of this research.

The next stage of the review was to read the complete texts of potentially relevant studies, applying inclusion/exclusion criteria. To facilitate the application of the criteria for inclusion and exclusion in the studies, a Microsoft Excel^{TM} spreadsheet was used.

The following inclusion criteria were used:

1) Studies from academia or industry;
2) Studies with practical scientific (empirical) or bibliographic data or experience reports;
3) Studies that addresses MPS.BR and agile methodology;
4) Studies in Portuguese or English.

As exclusion criteria were adopted:

1) Studies merely based on expert opinion, without being supported by a practical or bibliographic study or a report of an experience;
2) Studies in the format of an editorial, foreword, abstract, interview, news, poster and so forth.

At the end of this stage, 21 studies were included that address the MPS.BR model together with agile methodologies. The absence of inclusion criterion number 3 and the occurrence of exclusion criterion number 2 were the most frequently instances for excluding studies. When this stage was completed, we moved on to extracting data as described below.

C. *Data Extraction*

Data from 21 studies were extracted and analyzed, including: title, publication year, author, type (article, undergraduate monograph, post-graduate dissertation or Master's dissertation), publication source, where the research was conducted, research method (case study, experience report, survey, experiment, action research, ethnography, and literature), research goal, agile method addressed, MPS.BR levels involved, and the benefits and limitations of using MPS.BR and agile methodologies. Data from each study were copied to an Excel spreadsheet, to aid referencing during the stage of synthesizing the result.

IV. Results and Synthesis

The studies included 12 articles, 1 Master's dissertation, 3 post-graduate dissertations, 4 undergraduate monographs, and 1 undergraduate traineeship report. Fig. 1 shows the corresponding percentages. The articles were written at the following levels: 1 by post-graduate students, 1 by an undergraduate student, 4 by students of various levels, 2 by industry professionals, and 4 by students and professionals.

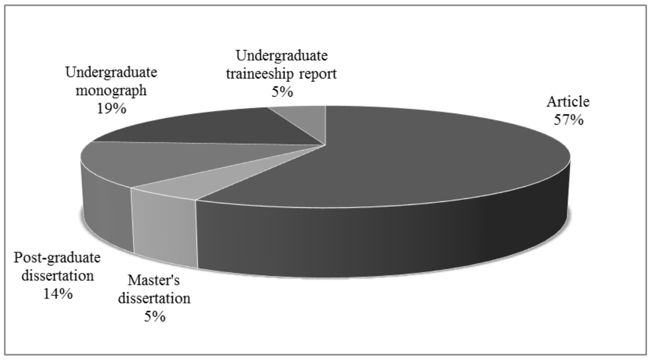


Figure 1. Type of studies.

The sources where the studies were published or produced are shown in Table 3. Note that articles published in 7 conferences, 1 magazine, and academic studies produced in 7 institutions of higher education were included.

TABLE III. Source of Studies

Type	Source	Study	No.
Article	Brazilian Symposium on Software Quality (SBQS)	[11][12] [13][14]	4
	Annual Workshop on SPI (WAMPS)	[15][16]	2
	Informatics Students Meeting of Tocantins (ENCOINFO)	[17]	1
	Innovations Week in Information Systems (SIS^{2}INFO)	[18]	1
	Regional Seminar on Informatics (SRI)	[19]	1
	Scientia Plena Magazine	[20]	1
	Symposium on Computational Mechanics (SIMMEC)	[21]	1
	Workshop on Companies (W6-MPS.BR)	[22]	1
Master's dissertation	Pernambuco University (UPE)	[23]	1
Post-graduate dissertations	Federal University of Sergipe (UFS)	[24]	1
	Pontifical Catholic University of Paraná (PUCRS)	[25]	1
	State University of Londrina (UEL)	[26]	1
Undergraduate monographs	Passo Fundo University (UPF)	[27][28]	2
	Santa Cruz do Sul University (UNISC)	[29]	1
	UniSEB University Center	[30]	1
Undergraduate traineeship report	State University of Londrina (UEL)	[31]	1
Total			**21**

Fig. 2 shows the percentage of studies with respect to the research method. Note that 9 of the 21 studies used Bibliographic Research. The 6 empirical studies used the following methods: 4 Case Studies, 1 Action Research, and 1 Survey. Six Experience Reports were also included.

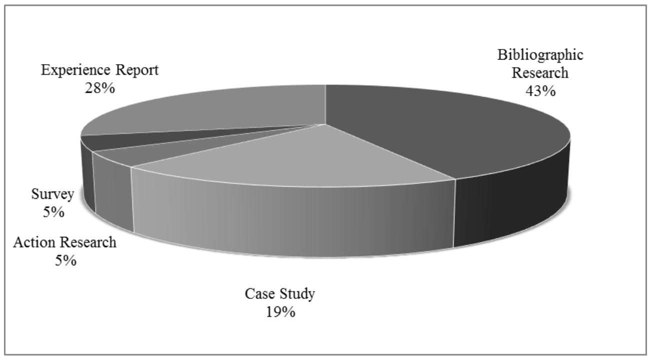


Figure 2. Research methods of studies.

As reported to the year, it was noted that 2010 was the year in which most studies were produced on the subject, as illustrated in Fig. 3. The number of studies has been growing every year since 2008, but in 2011 there was a small reduction. The geographical distribution of studies by State was as follows: 4 from Paraná, 4 from Rio Grande do Sul, 3 from Pernambuco, 3 from Rio de Janeiro, 2 from Minas Gerais, 2 from Sergipe, 2 from São Paulo, and 1 from Santa Catarina.

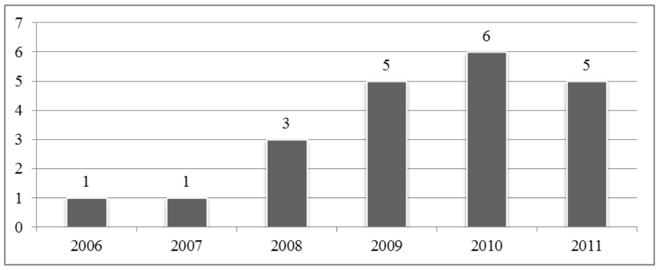


Figure 3. Year of studies.

In Fig. 4, the agile methodologies found in the studies are shown. Note that the Scrum methodology was the most used, being addressed in 17 studies (81%). In some studies, Scrum was used in combination with another methodology. The agile practices most addressed were Daily Meetings and Development in Sprints (in 13 studies), followed by Product Backlog Elaboration (in 11 studies), Sprint Review Meeting (in 9 studies), Sprint Planning Meeting and Retrospective (in 8 studies).

Fig. 5 shows the number of studies related to the levels of the MPS.BR model. The studies focus most on the initial levels (G and F). Although smaller, the number of studies that cites the other levels (A to E) was similar. The most addressed processes were Requirements Management (in 19 studies), Project Management (18), Quality Assurance (10), Measurement (9), Configuration Management (8), and Acquisition (6). The processes of Project Portfolio Management, Reuse Management, and Development for

Reuse were the least discussed (2 studies). The remaining processes were discussed in 3-5 studies.

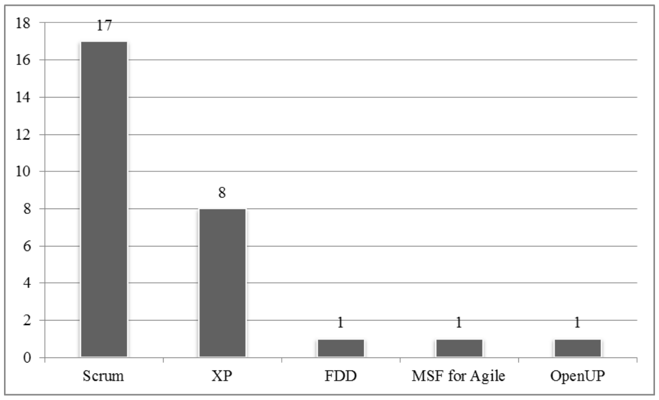


Figure 4. Agile methodologies addressed.

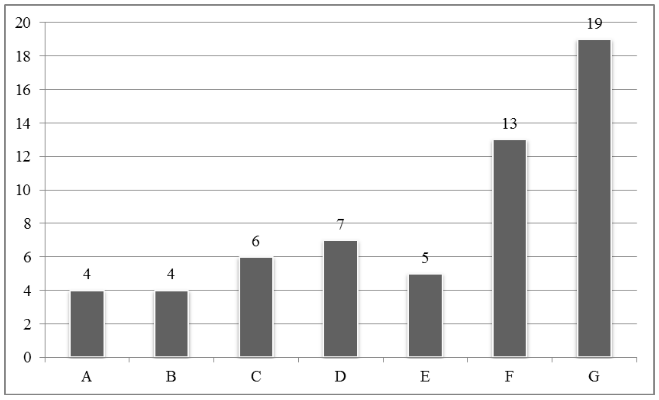


Figure 5. Number of studies by MPS.BR Levels.

Next, the benefits and limitations of MPS.BR together with agile methodologies are discussed.

A. *Benefits of using MPS.BR and Agile Methodologies*

At this stage of the review, the benefits were analyzed, as per how they were cited by the authors in the studies. Oliveira [29] found that the use of Scrum practices, such as maintaining the product backlog, satisfies many of the expected results from the processes of Requirements Management and Requirements Development, which correspond to levels G and D, respectively.

The study of Oliveira, Guimarães, and Fonseca [22] reports the experience in a company while the MPS.BR model together with Scrum and XP methodologies was being implemented, and notes the following benefits:

- Significant improvements with regard to team performance and the quality of final product.
- Indicators defined for the processes of Project Management and Requirements Management, such as productivity indicators, percentage of rework and percentage of deviation from predicted vs. actual, provided important feedback to the team and created goals to be achieved.
- Indicators also support decision making and create an atmosphere of continuous improvement.
- Indicators of Configuration Management ensured that certain practices were followed, providing greater control of the versions generated and continuous integration.
- Practices of Quality Management, such as audits, ensured the institutionalization of the development process and the quality of work products.

The study also considers that, when problems are identified, those responsible for Quality Management must submit proposals for solutions and improvements, and monitor the deliberations until completion.

According to Santesso [26], the combination of Scrum and MPS.BR proved to be satisfactory and feasible. Silva and Denardi [18] observed that the use of Scrum practices might bring quick results and with quality in the processes, in order to achieve the maturity levels of MPS.BR model. For Mancine [30], agile methods, in particular Scrum, were able to streamline the development processes.

Begnini [27] claims that the use of MPS.BR model can also be combined with XP agile methodology, bringing benefits to the company that aim to produce software with quality and greater agility.

In the study of Osawa [31], Scrum compatibility with the expected results of MPS.BR processes at level G was highlighted.

Considering the benefits mentioned by the studies, the use of agile methodologies with MPS.BR model succeeded in bringing improvements to organizations, which aim to produce software with agility and quality. However, the authors also pointed out limitations and challenges, which are discussed below.

B. *Limitations of using MPS.BR and Agile Methodologies*

Teixeira [28] concludes that Scrum practices alone is not able to meet all the requirements of MPS.BR, thus requiring the use of additional practices, as metaphor, planning game, pair programming, study documents, develop the model, and other, from other agile methodologies, such as XP and FDD. Several studies, such as [11][23][28], reported that one agile approach alone is not sufficient to achieve the maturity levels, and thus require some adjustments. In Silva, Magela, Santos, Schots, and Rocha [16], a combination with Unified Process and Scrum was proposed, showing that organizations can combine different approaches (agile and traditional) in order to comply with MPS.BR.

According to Stanga [19], to use agile practices of XP with the MPS.BR model some adjustments should be made to the project team, especially to aid Requirements Management. The study notes that a formal way of recording and monitoring requirement is necessary, but it does not offer a solution. The use of a tool for agile project management could help in this task. Some teams record manually the requirements in a spreadsheet or other document.

Begnini [27] notes that XP does not aid some processes of the MPS.BR model, because it does not prioritize the documentation of software developed nor the development and management of reusable components. Documentation and the production of an objective insight are challenges to which MPS.BR and agile methodologies are open. Agile teams should to define a minimum of essential documentation, in other hand, MPS.BR evaluators should understand agile values and be open to other forms of documentary representation.

When Salgado et al. [13] report the experience of implementing new processes adherent to MPS.BR level C, using Scrum practices, the main difficulties were presented:

- Discussions about process improvement can divert the focus of agile practices, such as retrospective meetings, e.g., making these meetings very long;
- All team members, including the Product Owner, must participate in the meetings, to provide communication and visibility of the Sprint status;
- Difficulty in estimating the size and time required to perform a certain activity, causing project delays;
- Team members should have a heterogeneous profile, thus avoiding a high turnover of team members.

Given the limitations observed in the studies, although it is possible the use of MPS.BR together with agile methodologies, there is the need to use additional practices, especially with respect to documentation and the metrics of storage.

V. Conclusion And Future Work

This article conducted a study, through a systematic review, on the MPS.BR model with agile methodologies, thereby aiming to contribute to quality improvements in both the development process and in the final software product.

In accordance with the included studies and the benefits pointed therein, the adoption of MPS.BR together with agile methodologies is feasible, mainly for the initial levels (G to D, except processes related to acquisition and reuse). They report that agile practices, which enable rapid improvements and significant quality in the processes and products, are needed to achieve maturity levels. However, the studies also pointed out limitations, such as the fact that agile methodologies could not completely satisfy the highest MPS.BR levels (C to A), thus requiring other practices, such as adjustments in the team, the representation of explicit knowledge and storage. These limitations can make it difficult to apply agile methodologies, and their benefits, in organizations. This demands alternatives that overcome mainly the problem of documental evidence.

Regarding the limitations of this review, the fact of not having performed a quality assessment of the studies does not allow an analysis of the strength of the results found. All studies underwent some review process, but this is not sufficient to provide a high level of quality. Another possible limitation is as to the coverage of the studies. Even though automatic and manual searches on major sources and indexing mechanisms were conducted, it is possible that relevant studies were not included, mainly studies produced in educational institutions, not published in journals or conferences. Studies produced from 2012 onwards were not included. Possible biases introduced throughout the process of study selection and data extraction are also considered as limitations. However, all the stages were performed by two researchers, and then revised by two other researchers who are knowledgeable about the area. The approach of MPS.BR gave this study a local scope (Brazilian context), but the authors undertook a systematic review on the benefits and limitations of CMMI and agile methodologies, as their scope was more global. The results will be presented in a future article.

The research presented in this paper may contribute to the academic area, since it presents an initial mapping of the studies conducted with respect to the issue addressed, as well as to organizations that focus on improving software development processes and adherence to best practices in order to ensure the quality of the software they develop. As a suggestion for future work, we put forward:

- Analyzing the adoption of agile methodologies with higher levels of MPS.BR model, aiming to find the possibility of smooth adaptation.
- The number of empirical studies found (28,5%), suggests the importance of more practical studies directed to the software industry, in order to meet its needs.

Acknowledgment

This study has the support of the Foundation for Science and Technology of Pernambuco State (FACEPE), through Process No. IBPG-0136-1.03/11.

References

[1] ISO/IEC 90003, “Software engineering - Guidelines for the application of ISO 9001:2000 to computer software”, 2004.

[2] ISO/IEC 12207, “Systems and software engineering - Software life cycle processes”, 2nd ed., 2008.

[3] SEI, Software Engineering Institute, “CMMI® for Development”, Version 1.3, Technical Report, CMU/SEI-2010-TR-033, November 2010.

[4] SOFTEX, Association for Promoting Excellence in Brazilian Software, “MPS.BR - Brazilian Software Process Improvement”, SPI General Guide for Software, December 2012.

[5] Agile Manifesto, “Agile Manifesto for Software Development”, 2001, http://www.agilemanifesto.org [retrieved: August, 2013].

[6] VersionOne, “State of Agile Development Survey Results”, 2011, http://www.versionone.com/pdf/2011_State_of_Agile_Development_Survey_Results.pdf [retrieved: August, 2013].

[7] H. Corbucci, A. Goldman, E. Katayama, F. Kon, C. Melo, and V. Santos, “Genesis and evolution of the agile movement in Brazil - a perspective from the academia and the industry”, Proc. 25th Brazilian Symposium on Software Engineering (SBES 11), IEEE Press, Sept. 2011, pp. 98-107, doi:10.1109/SBES.2011.26.

[8] K. Schwaber and M. Beedle, Agile Software Development with Scrum, Prentice Hall, 2002.

[9] K. Beck, Extreme Programming explained: embrace change, Addison-Wesley, 2000.

[10] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering”, Technical Report, School of Computer Science and Mathematics, Keele University, 2007.

[11] M. M. Arimoto, E. Murakami, V. V. Camargo, and M. I. Cagnin, “Adherence analysis of agile methods according to the MR-MPS Reference Model”, Proc. VIII Brazilian

Symposium on Software Quality (SBQS 09), SBC, Jun. 2009, pp. 249-263.

[12] E. Catunda et al., “Implementation of MR-MPS level F with Scrum agile practices in a software factory”, Proc. X Brazilian Symposium on Software Quality (SBQS 11), SBC, Jun. 2011, pp. 1-8.

[13] A. Salgado et al., “Applying an agile process for deploying software processes based on Scrum at Chemtech”, Proc. IX Brazilian Symposium on Software Quality (SBQS 10), SBC, Jun. 2010, pp. 351-358.

[14] C. Santana, A. M. L. Vasconcelos, and A. L. Timoteo, “Mapping of the Brazilian Model of Software Process Improvement (MPS.BR) for companies using Extreme Programming as development methodology”, Proc. V Brazilian Symposium on Software Quality (SBQS 06), SBC, Jun. 2006, pp. 130-143.

[15] R. Prikladnicki and A. L. C. C. Magalhães, “Implementation of maturity models with agile methodologies: an experience report”, Proc. VI Annual Workshop on SPI (WAMPS 10), SOFTEX, Oct. 2010, pp. 88-98.

[16] T. Silva, R. Magela, G. Santos, N. C. L. Schots, and A. R. Rocha, “Implementation of MR-MPS level F combining features of the Unified Process with Scrum practices”, Proc. VII Annual Workshop on SPI (WAMPS 11), SOFTEX, Oct. 2011, pp. 54-61.

[17] F. Szimanski, J. Albuquerque, and F. Furtado, “Implementing maturity and agility in a software factory using Scrum and MPS.BR level G”, Proc. XI Informatics Students Meeting of Tocantins (ENCOINFO 09), Lutheran University Center of Palmas, Nov. 2009, pp. 161-170.

[18] A. M. Silva and A. Denardi, “Using Scrum methodology to achieve MPS.BR level F”, Proc. XII Innovations Week in Information Systems (SIS^{2}INFO 11), UNIPAR, Oct. 2011, pp. 1-13.

[19] M. Stanga, “Integrating XP agile project with MPS.BR level G”, Proc. XX Regional Seminar on Informatics (SRI 11), URI, Sept. 2011, pp. 1-6.

[20] F. G. Silva, S. C. P. Hoentsch, and L. M. A. Silva, “An analysis of FDD and Scrum agile methodologies under MPS.BR quality model perspective”, Scientia Plena, vol. 5, Dec. 2009, pp. 1-13.

[21] T. M. R. Dias, G. F. Moita, M. P. Silva, B. Ferreira, and A. M. Silva, “Feasibility of software development based on the MPS.BR model with Extreme Programming methodology”, Proc. IX Symposium on Computational Mechanics (SIMMEC 10), UFSJ, May. 2010, pp. 1-10.

[22] A. C. G. Oliveira, F. A. Guimarães, and I. A. Fonseca, “Using agile methodologies to achieve MPS.BR level F in Powerlogic”, Proc. II Workshop on Companies (W6-MPS.BR), SOFTEX, Oct. 2008, pp. 1-8.

[23] C. Santana Júnior, “Evaluation of the Use of Agile Methodologies in the Context of Software Quality Models”, Master’s Dissertation in Computer Engineering, Pernambuco University (UPE), 2008, unpublished.

[24] P. Silva Neto, “Case Study of a Software Project Management Process Based on Agile Methods and MPS.BR Model”, Post-graduate Dissertation in Project Management of Information Technology, Federal University of Sergipe (UFS), 2010, unpublished.

[25] D. A. Nunes, M. A. F. Locks, and R. Hayashi, “To Check the Suitability of Using the Scrum Agile Methodology for MPS.BR Level G”, Post-graduate Dissertation in Software Project Management, Pontifical Catholic University of Paraná (PUCRS), 2011, unpublished.

[26] P. H. C. Santesso, “Using Scrum Agile Methods with MPS.BR Level G”, Post-graduate Dissertation in Systems Analysis, Design and Management with Emphasis on Business Intelligence, State University of Londrina (UEL), 2009, unpublished.

[27] M. M. Begnini, “An Analysis of XP From the Perspective of MPSBR Model, Undergraduate Monograph in Computer Science, Passo Fundo University (UPF), 2008, unpublished.

[28] M. Teixeira, “An Analysis of Scrum From the Perspective of MPSBR”, Undergraduate Monograph in Computer Science, Passo Fundo University (UPF), 2007, unpublished.

[29] J. S. Oliveira, “Scrum Methodology Applyed to Requeriment Managment and Development Process of MPS.BR Model”, Undergraduate Monograph in Information System, Santa Cruz do Sul University (UNISC), 2010, unpublished.

[30] M. H. Mancine, “Analysis of the Use of Scrum Methodologies and MPS.BR in the Real Case Study in Development of Commercial Applications”, Undergraduate Monograph in Computer Science, UniSEB University Center, 2011, unpublished.

[31] G. S. Osawa, “Analysis of the Implementation of Scrum in a Company with MPS.BR Level G”, Undergraduate Traineeship Report in Computer Science, State University of Londrina (UEL), 2009, unpublished.

Low-Overhead Profiling based on Stationary and Ergodic Assumptions

Stoyan Garbatov and João Cachopo
Software Engineering Group
INESC-ID Lisboa / Instituto Superior Técnico, Universidade Técnica de Lisboa
Lisbon, Portugal
stoyangarbatov@gmail.com and joao.cachopo@ist.utl.pt

***Abstract*—In the context of feedback-directed optimization solutions, the component responsible for collecting application behaviour data cannot afford to introduce any performance overheads, otherwise it undermines any optimization that is to be carried out. This work presents a new online solution for profiling object-oriented applications. The solution collects detailed information about accesses over domain instances and their fields, while introducing approximately zero overheads. This is accomplished by making assumptions about the stationary and ergodic properties of applications' run-time behaviour. The work has been validated with the TPC-W benchmark.**

Keywords-profiling; real-time monitoring; feedback-directed optimizations; performance; ergodic; stationary.

I. Introduction

The importance of profiling tools has been steadily increasing over the last decade. Profilers are essential for understanding the dynamic behaviour of programs. In the past, one of the most common use-case scenarios was to use a profiler for obtaining information about the resource usage of a given application, to identify performance bottlenecks. While the nature of the information supplied by profiling tools has not changed much over time, the spectrum of their possible applications has observed a significant widening. These applications include dynamic slicing, program invariants detection, program correctness and security checking, predicting data locality and just-in-time compiler optimizations, among others.

The widespread adoption of programming languages designed to execute on top of virtual machines played a major role in valorising profiling tools. Virtual machine architectures offer several properties that make them more desirable, from a software engineering point of view, than environments with statically compiled binaries. Some of these features include program portability, safety assurances, automatic memory and thread management, as well as dynamic class loading. As it can be seen from the work of Cierniak et al. [1], while these properties empower the programming model offered to users, they also cause overheads and contribute to obstruct many static program optimization techniques, making it harder to achieve good performance.

To counter these difficulties, a lot of research has been carried out, focusing on online feedback-directed optimization systems. These systems make use of techniques that seek to improve the performance of target programs by monitoring their run-time behaviour and, subsequently, by using this information for identifying and applying appropriate optimization measures. It is frequently the case to employ profiling tools for obtaining the necessary program behaviour information.

The main issue of profiling tools is that they are subject to two requirements that are practically impossible to satisfy simultaneously. The first requirement is that the profiler should provide as in-depth and detailed information as possible about the behavioural patterns exhibited by the program being monitored. The second requirement is that the monitoring should be carried out in a transparent, efficient and devoid of overheads manner that does not compromise program performance (or at least not significantly).

Independently of the way that a profiler operates (event or sampling-based), it invariably ends up disrupting the execution of its target programs, while data is being collected. These interruptions translate directly into overheads that penalize application performance. Furthermore, the fact that many profilers employ code instrumentation (the injection of additional code into the target) for achieving their goals not only slows down the execution of the program but can also change the way it operates, leading to scenarios where the application displays behaviour that would be otherwise impossible to observe, were it executing normally.

All of these factors contribute to make the task of striking an acceptable balance between performance overheads and information depth hard to achieve in practice. This is particularly true for online feedback-directed optimization systems, where the profiling is expected to be carried out in real-time, while the program is executing normally, so any "noticeable" performance overheads are not acceptable. On the other hand, the profiling information has to be sufficiently detailed to guide appropriately the optimization decisions that need to be made for improving the target system's performance.

While accounting for these considerations, the solution presented with this work consists in a system capable of monitoring the access patterns of object-oriented applications. The patterns are expressed in terms of the manipulations (read and write access operations) performed over domain instances and their fields in the execution contexts of the application methods/services that define the set of functionality offered to end-users. The novelty of the approach consists in the employment of stochastic and ergodic assumptions for the behaviour of target programs within their methods/services, making it possible to collect the access pattern information in an online fashion with minimal performance overheads while providing very high accuracy and data depth.

The article is organized as follows. Section II discusses related work. Section III describes the motivation behind our new proposal, followed by some assumptions that were necessary to be made in Section IV. Section V covers the implementation of the solution presented here. Section VI presents the benchmark that we used to evaluate the new proposal and discusses the results obtained. Finally, Section VII presents some concluding remarks.

II. RELATED WORK

The current trends in application development, software engineering, and hardware technology (such as the wide acceptance of programming languages operating in automatically managed virtual machine environments and the expansion of cloud computing, among many others) have contributed to create a great demand for solutions capable of continuously tracking the behaviour of dynamic systems and of applying performance optimization measures based on the gathered information. Adaptive optimization solutions such as these have been designated in the literature as online feedback-directed optimization (FDO) systems. As can be seen from the work of Arnold et al. [2], where an analysis of over 150 references related to online feedback-directed optimization solutions has been performed, there is a wealth of work done in this research area. Unfortunately, the great majority of these solutions have been developed to act as support for compilation and dynamic code generation optimizations, while non-compiler related literature is somehow scarce. Without intending to perform an in-depth and exhaustive analysis of the existing literature, some works on the topic of continuously tracking the behaviour of software systems shall be discussed next.

Smith [3] discussed the motivation and history of FDO techniques. The author presented three factors responsible for the importance of FDO, namely:

- FDO bypasses the restrictions imposed on static optimization approaches by making use of dynamic run-time behaviour information that is impossible to obtain statically;
- FDO makes it possible to adapt the optimization measures continuously, according to the observed changes in target application behaviour.
- Software systems can be made more flexible and easier to change through run-time binding.

From the analysis and discussion in [3], Smith argues in favour of performing optimizations based on run-time monitoring as well as accepting the notion of executables as mutable objects. Smith [3] and Arnold et al. [4] pointed out that the obstacles that need to be overcome to achieve an effective FDO are:

- Minimize or otherwise deal with the overhead introduced in the process of collecting behaviour information as well as when applying the necessary transformations over the target application for optimizing its performance;
- Being able to make informed decisions even when there is incomplete profiling data or that same information is subject to constant evolution.

In the context of virtual machine environments, it is possible to group the profiling-data collection mechanisms for FDO purposes into the following categories: run-time service monitoring, hardware performance monitors, sampling, and program instrumentation. The solution developed here belongs to the program instrumentation class of approaches. As such, the other categories shall be only referred to briefly.

Run-time service monitoring approaches track the state of the run-time services offered by the subjacent virtual machine. This is usually done for identifying temporal locality usage patterns that can be exploited for optimizations. Several applications of these techniques include dynamic dispatching, hash-codes, and synchronization. It is noteworthy that the memory management systems are particularly rich sources of data for FDO, covering information about allocation trends, heap usage and garbage collection.

Hardware performance monitoring collects data provided by specialized microprocessor hardware for guiding optimizations. There has been a multitude of FDO approaches developed to use such information but their integration into production-ready VMs has been limited.

With sampling approaches, the profiler seeks to collect a representative (as opposed to exhaustive) sub-set of observations for a given category of events. By varying the portion of events that get observed, sampling approaches can control the amount of overhead being introduced into the application that is being monitored. Nevertheless, as in all monitoring approaches, sampling techniques need to strike a balance between low overhead and collecting enough behaviour information to be considered useful.

The injection of extra instructions for collecting behaviour information into the target system is the basis for program instrumentation profiling approaches. These techniques are very flexible in their usage and can provide a wide range of behaviour data. Their main issue consists in the performance overhead caused by the need to execute the instrumented code. As such, most existing solutions attempt to minimize the overheads without compromising too much the depth of the profiled information.

Arnold and Ryder [5] developed a framework for low overhead instrumentation sampling supporting multiple types of profiling. The framework employs code duplication and compiler-inserted counter-based sampling to enable changes between the instrumented and original version of the target code at run-time. The amount of overhead to be introduced is adjustable at run-time, by varying the ratio of execution between instrumented and non-instrumented versions of the code. The authors achieve this by keeping in memory two versions of all methods that have been modified for profiling purposes. One of them is the instrumented version that performs the monitoring measurements while the other contains the original code version with a small preamble that determines if the fast or slow version of the method should be executed, depending on the current conditions.

Another software instrumentation system, called Pin, is the one developed by Luk et al. [6]. The tools offered by the system are written in C/C++ and support portable, transparent and efficient instrumentation. While the tools are mostly architecture independent, they can provide architecture specific information when required. The systems uses dynamic compilation for instrumenting applications at run-time. Pin employs several techniques for achieving high operation efficiency while collecting data. These include register re-allocation, inlining, liveness analysis and instruction scheduling. The authors evaluate their systems against other similar purpose solutions, such as Valgrind [7] and DynamoRIO [8] and demonstrate it is capable of offering better performance while providing similarly detailed levels of information.

III. Motivation

The solution presented here was developed for the purpose of supplying the input data necessary for the access pattern analysis and prediction techniques developed in [9-12]. These techniques use stochastic models for analysing and predicting, with a high degree of confidence, the domain data access patterns performed by target object-oriented applications. The models employed for this purpose are Bayesian Inference, Criticality Analysis and Markov Chains. The issue that was detected with these techniques resides in the performance overheads introduced by the process of collecting the input data necessary for the stochastic models. While the overhead is not very significant when evaluated in a single-threaded environment, where the average performance loss is in the order of 3% to 8%, when considered in a multi threaded environment, the performance loss observed is about 50%, which is absolutely unacceptable for any sort of real-time continuous application behaviour monitoring. Since the above techniques had been developed with the intent of supplying with information online feedback-directed performance optimization solutions, it is mandatory that they do not incur any noticeable performance overhead, otherwise their usefulness is compromised.

The new monitoring solution presented here was designed to deal with this issue. The reasoning behind the solution is as follows. It is very hard, if not impossible, to model and predict accurately the behaviour of an application as a whole, over long periods of time. The workload of (most) applications evolves continuously, as a function of external stimuli (such as client requests) and, apart from very specific scenarios or relatively short periods of time, it is impossible to know, a priori, the sequence of inputs/client-requests that will be issued at a given moment. This is what makes programs behave as non-stationary processes.

The stationarity of a process can be pictured intuitively as the absence of any drift in the set of realisations that defines its behaviour as time proceeds. From a mathematical point of view, this means that the probability distribution and density functions that describe such a process are unchanged by a shift in the time scale. They are applicable now and will remain so for all time.

The constantly evolving workload of applications makes it necessary to monitor them continuously, if a precise view of their behaviour is to be had. Furthermore, the monitoring has to be performed in a lightweight manner, otherwise it will introduce inadmissible performance overheads. The combination of these two factors demands an access-pattern analysis solution capable of delivering detailed information about application behaviour, which is expressed in terms of domain data manipulations being performed, without compromising program performance.

IV. Assumptions

Several assumptions had to be made to reach a viable solution that achieves these goals. The first assumption is that the workload of an application can be described entirely by the ratio of the invocation frequencies of the methods/services that define the functionality offered by that particular application to end-users.

The second assumption is that programs behave as stationary processes [13] within the execution contexts of their methods/services. The nature of the behaviour displayed when executing a particular method should not change significantly over time, as long as its implementation remains the same. There are several factors that make this reasonable to assume. The encapsulation and modularity properties observed in (well-designed and implemented) object-oriented applications allow their methods to display functionality that is well defined and contained. This makes it highly unlikely to observe a broad range of different access-pattern behaviours when executing a particular method, independently of small shifts and variations that can occur when operating over different arguments.

The third and last assumption is that the operation of application methods displays ergodic properties. A process is ergodic [14] if it is stationary and, furthermore, if it is possible to extract its statistical descriptors from realizations that cover a single finite period of time. Intuitively, it may be said that the realisations obtained from this time period are "typical" of all the possible realisations, if the process is to be ergodic. In practice, it is not necessary for the methods to be strongly ergodic. It is enough to be able to extract the behaviour descriptors from a finite number of observations. This translates into being able to extract the typical access-pattern behaviour of a method from a limited number of invocations.

V. Implementation

A. *Compile-time*

There are two main components of the solution presented here - a code injection module and a data acquisition module. The first module employs the ASM byte-code manipulation toolkit [15] for injecting, at compile-time, code into target applications. This code invokes functionality in the data acquisition module. In particular, the injected code serves two purposes. The first consists in updating the information about changes in the execution context within which operations are taking place. By manipulating byte-code, the first instruction of all application methods or services (of interest) is defined to invoke a method responsible for updating the profiler state information that a new execution context has been initiated. Similarly, the last instruction before returning (or otherwise terminating the current execution context) calls functionality that clears up the profiler state information about the no-longer-active context.

The second task of the code injection module is to replace all accesses to domain instances (and their fields) with the invocation of a distinct method, depending on the type of access being replaced (read or write operation). This method is responsible for resolving the surrounding execution context as well as for updating the statistical information about the domain data access that is to be performed within the context that has been identified.

B. *Run-time*

The second module is responsible for collecting the behaviour data displayed by the target application, in terms of the domain data access patterns performed while it is executing. To obtain high-precision data without introducing performance overheads, the gathering is performed in two stages - a learning period and a steady-state period.

The first stage defines a period during which the monitoring system builds a detailed profile for all methods and/or services (of interest) of the target application. Each of the profiles assembled in this stage contains the typical access-patterns that are performed, per invocation of the corresponding method or service. The patterns are described in terms of the frequency of accesses performed over domain instances and their fields. Consequently, at the end of this stage, the information collected within the profiles indicates the number of read and write operations that are usually observed to be performed over domain data types when executing the associated methods/services (e.g. MethodY {DomainDataA.Field2 = 54 reads; DomainDataC.Field7 = 17 writes}, MethodX {DataD.Field3 = 7 reads}, etc.).

While the target application is executing in profile-building mode, it operates with an enriched version of its byte-code that contains the calls to the context updating functionality, as well as statistical behaviour collection. The necessity to execute all this extra code leads to significant performance overheads. That is why the learning stage proceeds only until a representative profile has been built for all the noteworthy methods. Once this has been accomplished, the application can move on to the next stage.

In the second stage, the only injected code that is kept in the target application is the one responsible for keeping track of the changes in execution contexts. Behaviour data is no longer collected about the access patterns that are effectively being practiced by the application. This allows the target system to operate with practically unperturbed performance, when compared to its original version, as shall be seen and demonstrated in the results and evaluation section. The extremely low overhead makes it possible for the application to operate normally, while the monitoring system solution keeps its profiling data up-to-date.

The question that remains is how does the profiler system update the overall access-pattern behaviour information, if the only application aspect that it keeps track of is the change in execution contexts. If the program behaviour, at method level, is stationary (and does not drift significantly over time) then the method profiles built during the learning stage continue to provide a precise view of the behaviour displayed when executing those methods, as long as their implementation does not change. As such, whenever an updated overall view of the domain-data access patterns is necessary, the profiler determines the composition of the workload, based on the observed ratios of method/service invocations (e.g. MethodX = 1045 invocations, MethodY = 703 invocations, etc). The interval for which the workload is determined corresponds to the period of time from the previous update up to the moment when the new update is requested.

Once the workload has been identified, this information is used along with the individual method/service profiles for building an application-level view of the domain access patterns performed during that period. Simply put, the individual profiles are weighted by the workload ratio that their respective methods assumed in the workload, for that particular period of time.

It should be noted that if the implementation of a method does change, at some point in time, then it is necessary to revert the application back to stage one so that new and updated profiles can be built. Otherwise, there would be no guarantee as to the correctness of the access-pattern information generated by the profiler.

VI. RESULTS AND EVALUATION

The TPC-W benchmark [16] was used to evaluate the performance of the profiling solution presented here. The benchmark specifies an e-commerce workload that simulates the activities of an online retail store, where emulated clients can browse and order products from the site.

The TPC-W evaluation metric is the number of web interactions per second (WIPS) that the system can sustain. The benchmark execution is characterized by a series of input parameters. One of these defines the type of workload, which specifies the percentage of read and write operations that is to be simulated by the emulated browser (EB) clients. The workloads considered were Mix1 (95% read and 5% write); Mix2 (80% read and 20% write) and Mix3 (50% read and 50% write). The remaining configuration parameters were 10 emulated browsers; 300 seconds of ramp-up time; 1200 seconds for measurement interval, after the ramp-up time; 120 seconds of ramp-down time; 1k, 10k, and 100k book items in the database and think time of 0 seconds, ensuring that EBs do not pause in between requests.

All results were obtained as the average of 10 independent executions of the benchmark, for identical configurations. The EBs, database and the benchmark server were run on the same physical machine. The measurements were carried out with the benchmark running on a machine equipped with 2x Intel Xeon E5520 (a total of 8 physical cores with hyper-threading running at 2.26 GHz) and 24 GB of RAM. Its operating system was Ubuntu 10.04.3, and the JVM used was Java (TM) SE Runtime Environment (build 1.6.0 22-b04), Java HotSpot (TM) 64-Bit ServerVM (build

17.1-b03, mixed mode). The benchmark was run on top of Apache Tomcat 6.0.24, with the options set to "-server -Xms64m -Xmx$(heapSize)m -Xshare:off -XX:+UseConcMarkSweep GC -XX:+AggressiveOpts".

The throughput of the benchmark was evaluated for 14 different modes of monitoring. The operation mode designated as *BaseLine* corresponds to the TPC-W operating in pristine conditions, with its original implementation, without any byte-code manipulations. The newly developed profiler shall be referred to as the *Stationary* solution, while the old monitoring solution, that keeps track of domain access pattern occurrences continuously, shall be referred to as *NonStationary*.

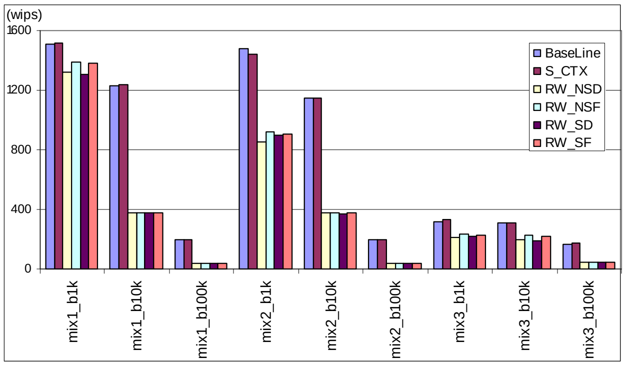
Two different approaches (orthogonal to the profiling solution in use) for storing data about changes in execution context shall be considered as well. The first of these approaches, which shall be called *Deep*, maintains information about the sequence of context changes that led to the currently active one. Such an approach makes it possible to know the exact sequence of method invocations that preceded any given point in the execution of the program. The alternative approach, called *Flat*, only keeps track of the currently active execution context, independently of the execution flow that might have been displayed by the program to reach it.

To get a better grasp of how the *Stationary* and *NonStationary* solutions behave, three further variants are taken into account, based on the types of accesses over domain data that are tracked. These are read-write, write-only and read-only modes. The list of the 14 different modes of monitoring evaluated here is:

- BaseLine - vanilla version of TPC-W
- S_CTX - *Stationary* solution in context-only mode;
- RW/WO/RO_NSD - *NonStationary* solution with *Deep* context tracking in Read-Write, Write-Only and Read-Only modes;
- RW/WO/RO_NSF - *NonStationary* solution with *Flat* context tracking in Read-Write, Write-Only and Read-Only modes;
- RW/WO/RO_SD - *Stationary* solution under profile-building mode, with *Deep* context tracking in Read-Write, Write-Only and Read-Only modes;
- RW/WO/RO_SF - *Stationary* solution under profile-building mode, with *Flat* context tracking in Read-Write, Write-Only and Read-Only modes.

A total of 126 distinct benchmark configurations were evaluated (14 operation modes, 3 workload types and 3 data-base sizes). Additionally, every configuration was executed 10 times, independently of previous runs, to provide a more comprehensive view of the behaviour displayed by the system. Taking this into account, along with the fact that a single execution of the benchmark takes approximately 15min (14min benchmark execution and 1min for Tomcat reboot, benchmark redeploy and database refresh), the results presented in this section took a total of 315h to generate.

The WIPS achieved by the *BaseLine*, *Stationary* in context-only mode and the Read-Write of the *Stationary* and *NonStationary* can be seen in Figure 1 (top), while the Write-Only and Read-Only variants Figure 1 (bottom). Every group of bars corresponds to a particular benchmark configuration in terms of workload (mix1, mix2 and mix3) and database size (1k, 10k and 100k book instances).



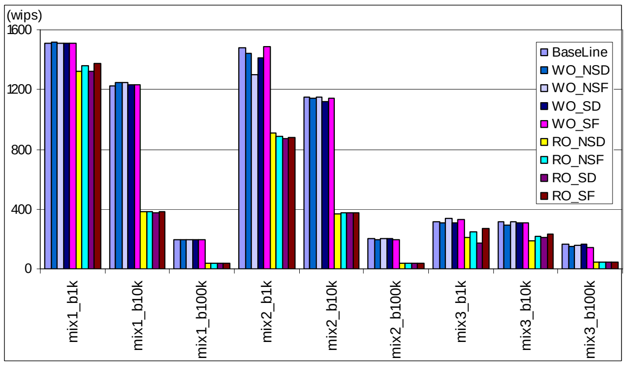


Figure 1. WIPS - baseline, stationary and non-stationary monitoring

From the analysis of these results, several remarks can be made. The throughput displayed when TPC-W is operating with *Stationary* in context-only mode appears to be very similar to the one when the benchmark is operating in its original version. When the benchmark is operating with *Stationary* (ReadWrite and ReadOnly) in profile-building mode as well as with *NonStationary* (ReadWrite and ReadOnly), the throughput observed is significantly lower than the one of *BaseLine*. The throughput displayed by the benchmark when the monitoring solutions are only tracking write-access operations over domain data is very similar to the BaseLine. The last two observations confirm what was already to be expected, namely that the factor that contributes most for the performance overheads observed when profiling a program is the tracking of read-access operations. Adding a fixed overhead to many short duration operations is bound to cause a greater impact than adding the same overhead to few long-duration operations.

It is interesting to note that for both *Stationary* and *NonStationary* approaches, whenever they are operating with *Flat* context-tracking mode, the WIPS achieved are slightly but consistently better than their counterparts with *Deep* context-tracking mode.

A thorough comparison of the relative throughput difference between all 14 monitoring modes and the BaseLine can be seen in Table I and II. The relative throughput difference has been calculated as

$\left(\left(T_A - T_{BaseLine}\right)/T_{BaseLine}\right)\times 100$, in %, where T_A indicates the throughput achieved when TPC-W is operating with monitoring approach A. The average throughput difference, per approach, is displayed with a shaded background.

TABLE I. THROUGHPUT DIFFERENCE (%), ALL VS. BASELINE (R&W)

	BaseLine	S_CTX	RW_NSD	RW_NSF	RW_SD	RW_SF
mix1_b1k	0.00	0.51	-12.36	-8.06	-13.15	-8.57
mix1_b10k	0.00	0.63	-69.01	-69.12	-69.53	-68.95
mix1_b100k	0.00	-0.01	-81.45	-80.99	-81.50	-81.26
mix2_b1k	0.00	-2.49	-42.11	-37.75	-39.28	-38.70
mix2_b10k	0.00	-0.22	-66.85	-66.91	-67.44	-67.10
mix2_b100k	0.00	-0.34	-82.10	-81.73	-82.13	-81.53
mix3_b1k	0.00	5.16	-32.85	-25.43	-30.87	-27.70
mix3_b10k	0.00	-1.46	-36.17	-28.74	-40.24	-29.25
mix3_b100k	0.00	3.45	-73.08	-72.71	-72.81	-72.04
avg	0.00	0.58	-55.11	-52.38	-55.22	-52.79

The analysis of these results indicates that the performance of the benchmark, when the Stationary approach is operating in context-only mode (S_CTX) is practically identical (0.58% difference) to the one of the original version of TPC-W (BaseLine). While it would be logical to expect that the S_CTX should display throughput that is strictly less than the one of the BaseLine, the few configurations where the opposite is observed can be (eventually) explained by the fact that the byte-code manipulations performed over the benchmark, for keeping track of the changes in execution contexts, allowed the just-in-time compiler of the JVM to perform some further optimizations that would be otherwise unable to apply. Another (possibly more likely) explanation for this phenomenon would relate to the intermediate-to-high measurement uncertainty observed for the two configurations where the S_CTX performs better than BaseLine (Mix3 with b1k and b100k).

TABLE II. THROUGHPUT DIFFERENCE (%), ALL VS. BASELINE (READ-ONLY AND WRITE-ONLY)

	WO_NSD	WO_NSF	WO_SD	WO_SF	RO_NSD	RO_NSF	RO_SD	RO_SF
mix1_b1k	0.47	0.25	0.22	-0.03	-12.33	-9.75	-12.47	-8.59
mix1_b10k	1.69	1.29	0.68	0.52	-69.06	-68.57	-69.18	-68.92
mix1_b100k	-0.08	-0.01	0.25	-0.24	-81.63	-81.49	-81.73	-81.06
mix2_b1k	-2.19	-11.88	-4.08	1.00	-38.27	-39.82	-40.94	-40.48
mix2_b10k	-0.52	0.44	-2.20	-0.10	-67.86	-67.18	-67.17	-67.07
mix2_b100k	-0.66	-0.17	-0.15	-0.55	-82.18	-81.61	-82.02	-81.90
mix3_b1k	-4.01	5.61	-3.42	4.02	-33.10	-23.13	-45.84	-14.76
mix3_b10k	-5.32	-0.26	-1.76	-1.40	-38.98	-30.35	-33.02	-24.80
mix3_b100k	-9.60	-3.25	0.53	-15.55	-73.18	-72.57	-72.70	-72.71
avg	-2.25	-0.89	-1.10	-1.37	-55.18	-52.72	-56.12	-51.14

The throughput displayed by the *Stationary* solution in profiling-mode is very similar to the *NonStationary* approach, across all evaluated configurations. This was to be expected since both approaches perform very similar tasks, in those operation modes. On the average, the throughput that the TPC-W can maintain while operating with *Stationary* profile-mode or any of the *NonStationary* variants is from 51% to 56% lower than the throughput of the unmodified benchmark.

The last performance aspect that can be appreciated, based on these results is the effect of the *Deep* and *Flat* context-tracking modes. As could be seen from Figure 1 and can now be confirmed numerically, the *Flat* context-tracking mode allows for small but consistent performance improvements. These are most noticeable for the Read-Write and Read-Only variants of the monitoring solutions and range from 3% to 5%.

VII. CONCLUSION

This work presented a new solution for profiling the behaviour of object-oriented applications, in terms of the access-patterns performed at run-time over domain data. By making certain assumptions about the stationary and ergodic properties of the run-time behaviour of object-oriented applications, the new solution can provide detailed and continuously updated information about the effectively practiced domain-data access-patterns, by the target application, without introducing any noteworthy performance overheads. This feature allows the newly developed solution to monitor any application in real-time, while the target system is operating in steady-state.

The solution was evaluated on the TPC-W benchmark, against multiple variants of previously existing solutions. It was possible to demonstrate that the new approach reduces the performance overheads of previous alternatives from an average of 55% down to approximately zero, while providing the same degree of information.

ACKNOWLEDGMENT

This work was partially supported by national funds through FCT – Fundação para a Ciência e a Tecnologia, under project PEst-OE/EEI/LA0021/2013, as well as by FCT (INESC-ID multiannual funding) PIDDAC Program funds and by the Specific Targeted Research Project (STReP) Cloud-TM, which is co-financed by the European Commission through the contract no. 257784. The first author has been funded by the Portuguese FCT under contract SFRH/BD/64379/2009.

REFERENCES

[1] M. Cierniak, M. Eng, N. Glew, B. Lewis and J. Stichnoth, 2003, The Open Runtime Platform: A Flexible High-Performance Managed Runtime Environment, Intel Technology Journal, 7, (1), pp. 5-18.

[2] M. Arnold, S. Fink, D. Grove, M. Hind and P. Sweeney, 2005, A survey of adaptive optimization in virtual machines, Proceedings of the IEEE, 93, (2), pp. 449-466.

[3] M. Smith, 2000, Overcoming the challenges to feedback-directed optimization, ACM SIGPLAN Notices, ACM, Vol. 35, pp. 1-11.

[4] M. Arnold, M. Hind and B. Ryder, 2002, Online feedback-directed optimization of Java, ACM SIGPLAN Notices, ACM, Vol. 37, pp. 111-129.

[5] M. Arnold and B. Ryder, 2001, A framework for reducing the cost of instrumented code, ACM SIGPLAN Notices, ACM, Vol. 36, pp. 168-179.

[6] C. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. Reddi and K. Hazelwood, 2005, Pin: building customized program analysis tools with dynamic instrumentation, ACM SIGPLAN Notices, ACM, Vol. 40, pp. 190-200.

[7] Valgrind, http://valgrind.org/ [accessed 10 May 2013].

[8] DynamoRIO, http://www.dynamorio.org/ [accessed 10 May 2013].

[9] S. Garbatov, J. Cachopo and J. Pereira, 2009, Data Access Pattern Analysis based on Bayesian Updating, Proceedings of the First Symposium of Informatics (INForum 2009), Lisbon, p. Paper 23.

[10] S. Garbatov and J. Cachopo, 2010, Importance Analysis for Predicting Data Access Behaviour in Object-Oriented Applications, Journal of Computer Science and Technologies, 14, (1), pp. 37-43.

[11] S. Garbatov and J. Cachopo, 2010, Predicting Data Access Patterns in Object-Oriented Applications Based on Markov Chains, Proceedings of the Fifth International Conference on Software Engineering Advances (ICSEA 2010), Nice, France, pp. 465-470.

[12] S. Garbatov and J. Cachopo, 2011, Data Access Pattern Analysis and Prediction for Object-Oriented Applications, INFOCOMP Journal of Computer Science, 10, (4), pp. 1-14.

[13] G. Lindgren, 2012, Stationary Stochastic Processes: Theory and Applications: Chapman and Hall/CRC.

[14] P. Walters, 1982, An Introduction to Ergodic Theory: Springer.

[15] ASM, http://asm.ow2.org/, [accessed 10 May 2013].

[16] W. Smith. TPC-W: Benchmarking An Ecommerce Solution. Intel Corporation, 2000.

A Tracking and Visualizing System of Memory Usage along to C Source Programs

Kyoko Iwasawa
Computer Science dept.
Takushoku University
Tokyo, Japan
kiwasawa@cs.takushoku-u.ac.jp

Takuhiro Okamuira
COMSOFT Co, Ltd.
Tokyo, Japan
shnfkrm@live.jp

Abstract— **Our tracking system shows the situation of memory usage of C programs. Its information includes both stack area of local data and dynamic allocated area. The system shows the results off line. First of all, it inserts recording statements to the memory allocation and release of user programs in order to make a log file of memory usage. After the execution of modified user program, the system analyzes the log file to make the usage graph. Based on the graphical image, the user can find out where each memory event occurred on the C source program interactively. Therefore, the user can recognize the accurate location where the largest memory area was used, and find which memory allocation caused memory leak. These functions are efficient for embedded system, whose memory size is strictly limited. In this work in progress, we are attempting to show where user should insert *free* function call by using static data flow analysis.**

Keywords-C source program; memory usage; memory leak; tracking; visualizing;

I. Introduction

C program developers have to be concerned with the situation of memory usage of their programs. The amount of memory usage is influential in performance of the program, because using large memory often causes cache miss and paging. Also, amount size of memory usage is essential for embedded system [2]. However, it is difficult to know how much memory is necessary and where it should be decreased.

Although, generally, it is difficult to know that when memory usage becomes maximum size and whether there is leak area, this information is necessary to optimize C programs. Programmers need to know the relationship between source code and memory usage. There are some memory management tools for linux. One of them is valgrid [3], which is the multipurpose code profiling and memory debugging tool. It shows whether the memory leak occurred and finds invalid pointer use [4]. It shows the leaked memory address with process ID, so the users would have to look for that address on the allocated memory address list. And it has lots of functions, but does not mention with the size of memory usage.

Our system shows graphically the amount of memory usage in chronological order. For each point on the usage graph, user is able to know the source corresponding source statements interactively. In addition, the system corresponds dynamic allocated memory (i.e., *malloc*() and *calloc*() call) to its release (i.e., *free*() call) [1].

Section II describes the tracking and visualizing system and its output images. Section III shows detail of the log file. Section IV presents the algorithm to calculate the dynamic allocated area, and Section V concludes and describes the future work.

II. Outline Of The System

In order to measure accurately the size of memory usage, our system analyzes executing logs offline. Therefore, the system parses the target C programs and inserts output statements to record of memory events.

A. Process of Tracking and Visualizing

Figure 1 shows the analyzing process. First of all, the system picks up statements which cause memory allocation and release. The picked up statements are following four kinds of statements.

(1) Entry point of functions
(2) Return statement in functions
(3) Invoke *alloc* function
(4) Invoke *free* function

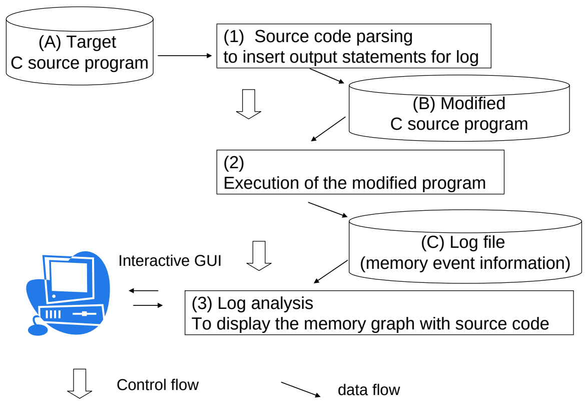


Figure 1: Process of the System

At point (1) and the point (2) the system inserts our prepared function call in order to write the size of local data on stack area, and to write when these local data are released. At point (3) and the point (4) the system replaces the library call with our prepared function call, which writes the information of dynamic allocated data to the file. Then the modified C source program (Figure 1: (B)) is generated. After the execution of the modified program the system can get the log file which includes the memory usage information. Finally, memory usage graph is displayed and users are able to know details of memory information.

B. *Output of the System*

After the system analyzes the log file, it visualizes the memory usage of C program. By restructuring the log data, memory usage graph is displayed off line as Figure 2. The upper part of the figure shows behavior of stack area usage, and the lower part of the figure shows the behavior of dynamic allocated heap area. Each bar means that the four kinds of memory events, which were described in the previous section and denoted by (1), (2), (3) and (4), occurred. The length of bar shows the size of memory used by program at that point.

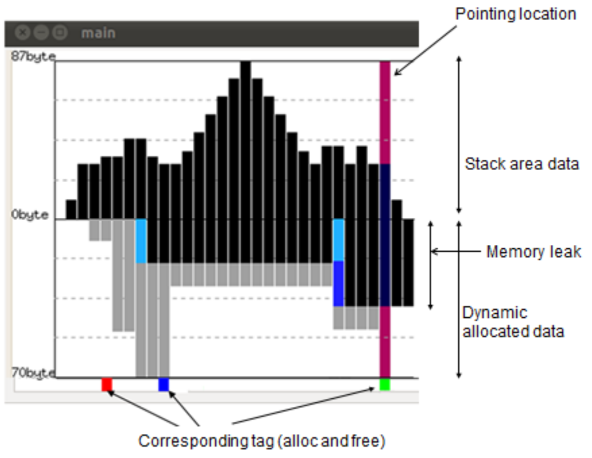
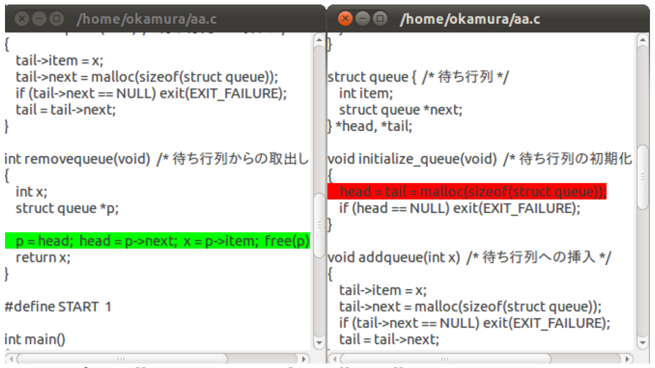


Figure 2: Memory usage graph



```
{
   tail->item = x;
   tail->next = malloc(sizeof(struct queue));
   if (tail->next == NULL) exit(EXIT_FAILURE);
   tail = tail->next;
}

int removequeue(void)  /* 待ち行列からの取出し
{
   int x;
   struct queue *p;

   p = head;  head = p->next;  x = p->item;  free(p)
   return x;
}

#define START  1

int main()
```

```
}

struct queue {  /* 待ち行列 */
   int item;
   struct queue *next;
} *head, *tail;

void initialize_queue(void)  /* 待ち行列の初期化
{
   head = tail = malloc(sizeof(struct queue));
   if (head == NULL) exit(EXIT_FAILURE);
}

void addqueue(int x)  /* 待ち行列への挿入 */
{
   tail->item = x;
   tail->next = malloc(sizeof(struct queue));
   if (tail->next == NULL) exit(EXIT_FAILURE);
   tail = tail->next;
```

(a) *free* call (b) *malloc* call

Figure 3: Source program

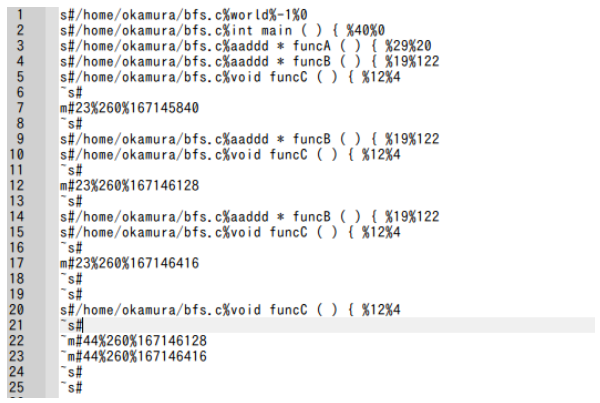
When user function is invoked, the size of local data area is added to usage size, and when return statement executes, these local data area is released, so the length of bar is shortened. When the C program has finished, its local data area is always cleared. On the other hand, heap area data, which was allocated by *alloc* function call, is released only by *free* function call. Consequently, if there is not enough *free* function call, memory leak will occur when program has finished.

When user clicks on a bar of the graph in Figure 2, the system shows the source program which caused this event. Figure 3 (a) shows the *free* function call of the source program, which corresponds to the clicked bar in the lower part. It shortened because the *free* function call releases memory area. A bar in the lower part lengthens, when *malloc* function call allocates memory area dynamically, as shown in Figure 3 (b). At the end of the execution, a bar in the lower part means memory leak. User might insert *free* function call, because the user can find out where this area is allocated and its identifier from the tags on the bottom of the graph (as seen in Figure 3:). If *free* function call released linked area (linked list or tree or graph structure, etc.), the system founds all *alloc* function call statements. The small tags on the bottom of the graph mean the correspondence *free* call to *alloc* call.

The bar in the upper part increases when user function call invoked, and then it shortens by return. As these bars in the upper part mean the stack area, when a program has finished, it always becomes zero. The total length of a bar means the size of memory, which user program was using at that point.

III. DATA IN THE LOG FILE

The detail of the logging data (shown as Figure 4:) is described in this section. In order to measure accurately the size of memory usage, user program is parsed and modified. User program is inserted the system function call to log the memory information.



```
s#/home/okamura/bfs.c%world%-1%0
s#/home/okamura/bfs.c%int main ( ) { %40%0
s#/home/okamura/bfs.c%aaddd * funcA ( ) { %29%20
s#/home/okamura/bfs.c%aaddd * funcB ( ) { %19%122
s#/home/okamura/bfs.c%void funcC ( ) { %12%4
~s#
m#23%260%167145840
~s#
s#/home/okamura/bfs.c%aaddd * funcB ( ) { %19%122
s#/home/okamura/bfs.c%void funcC ( ) { %12%4
~s#
m#23%260%167146128
~s#
s#/home/okamura/bfs.c%aaddd * funcB ( ) { %19%122
s#/home/okamura/bfs.c%void funcC ( ) { %12%4
~s#
m#23%260%167146416
~s#
~s#
s#/home/okamura/bfs.c%void funcC ( ) { %12%4
~s#
~m#44%260%167146128
~m#44%260%167146416
~s#
~s#
```

Figure 4: Log file

A. Local Data on the Stack Area

The size of local data on the stack area is accumulated the sum of all sizes of local data (variable, array, etc.), which are caused by nested user function call.

(1) Entry point of functions

The system inserts the function call to output file following five items to the log file.

- Tag (“s” means entry point of function)
- Directory path and file name
- Function name
- Line number of entry point of function in Source program
- Sum of local data size (byte) from static data declarations

(2) Return points of functions

When return statement is executed, system records the tag, which shows that the program returned from the function and its stack area data is released.

- Tag (“~s” means return point of function)
- Directory path and file name

B. Dynamic Allocated Data

The size of dynamic allocated data on the heap area is recorded by each *alloc* (*malloc* and *calloc*) function call and *free* function call. As a result, we can get the log file as Figure 4 shows. It includes the accurate and detailed memory usage situation. Each record is kept in the chronological order.

(3) Invoke *malloc* and *calloc* function

- Tag (“m” means *alloc* function call)
- Line number of *alloc* function call
- Size of allocated area (actual parameter of *alloc* call)
- Address of allocated area (return value of *alloc* function call)

(4) Invoke *free* function

- Tag (“~m” means *free* function call
- Line number of *free* function call
- Size of released area (system find out its own data → sec.4)
- Address of released area (actual parameter of *free* function call)

IV. CALCULATION OF DATA SIZE

Although we would like to know the leaked memory size and where the leaked memory was allocated on source code, some information of dynamic allocated data was lost when the program was finished. Therefore, the system function has to collect information while user program is running.

In the case of linked structure (list, tree and graph, etc.), it is difficult to accumulate released area by *free* function call. The parameter of *free* function is the pointer of the data, so there is no information as to the size of the data. Furthermore, *free* function released all of linked data, so that the system has to keep allocated memory information until it is released by using linked list that we call “Alloc List”.

A. Alloc Function Call

Our system function $ALLOC, which is invoked instead of the original *alloc* function, has the following three tasks.

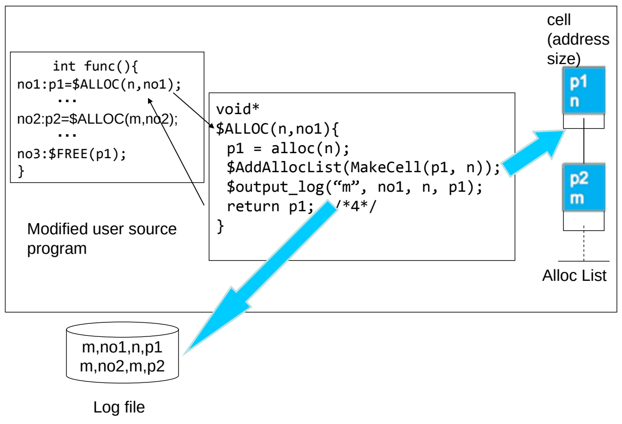
☐ To allocate memory by calling original *alloc* function

☐ To make the cell of Alloc List including the address and the size

☐ To write the log file (tag, address, size, line number of source program)

This function is given line number of source program and the data size to allocate as formal parameters. Figure 5 shows the process of $ALLOC. The system replaces the *alloc* function call to $ALLOC function call, and links $ALLOC function to modified program.

The cell of Alloc List has the address which is returned by *alloc* function and allocated memory size (byte), which is actual parameter of *alloc* function. Each cell is connected in chorological order as Alloc List.



```
    int func(){
no1:p1=$ALLOC(n,no1);
     ...
no2:p2=$ALLOC(m,no2);
     ...
no3:$FREE(p1);
}
```

```
void*
$ALLOC(n,no1){
  p1 = alloc(n);
  $AddAllocList(MakeCell(p1, n));
  $output_log(“m”, no1, n, p1);
  return p1;   /*4*/
}
```

Figure 5: $ALLOC function and Alloc List

B. Free Function Call

Our system function $FREE, which is invoked instead of the original *free* function call, has following the four tasks.

☐ To find address given as formal parameter from Alloc List.

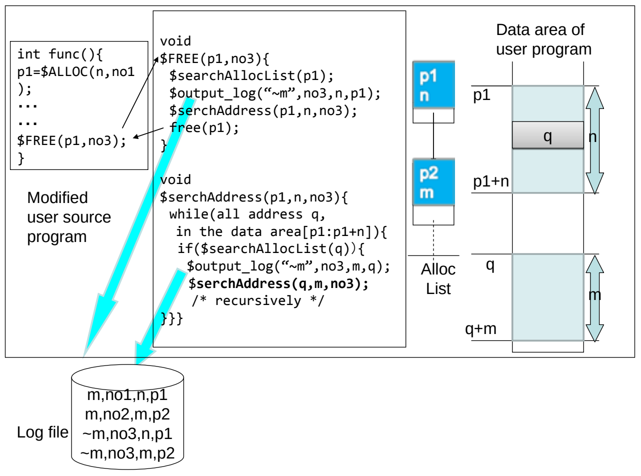
☐ To write log file (tag, address, size, line number)

☐ To search linked area to release recursively from Alloc List

☐ To release area by calling original free function

This function is given the address that is at the top of release area, and line number of source code as formal parameters. Figure 6 shows the process of $FREE function. The system replaces the *free* function call by the $FREE function call, and links $FREE function to modified program.

The system finds the address, which is the actual parameter of original *free* function call, from Alloc List, and writes the line number, size, and address. Then, the system has to find any linked area which would be released by the same *free* function call, so it searches any allocated address between given address and given address + size. If system finds out the address that is kept in Alloc List, it will search the next linked area recursively. While finding each area, system gets which alloc function call corresponds to this *free* function call.



```
int func(){
p1=$ALLOC(n,no1
);
…
…
$FREE(p1,no3);
}
```

```
void
$FREE(p1,no3){
  $searchAllocList(p1);
  $output_log("~m",no3,n,p1);
  $serchAddress(p1,n,no3);
  free(p1);
}

void
$serchAddress(p1,n,no3){
  while(all address q,
   in the data area[p1:p1+n]){
   if($searchAllocList(q)){
     $output_log("~m",no3,m,q);
     $serchAddress(q,m,no3);
      /* recursively */
}}}
```

```
m,no1,n,p1
m,no2,m,p2
~m,no3,n,p1
~m,no3,m,p2
```

Figure 6: $FREE function and Alloc List

V. Conclusion And Future Works

Our tracking and visualizing system shows the situation of memory usage of C programs. It includes both stack area of local data and dynamic allocated area. It makes a log file about memory usage by modified user program execution, and then analyzes the log file to make the memory usage graph. Based on graphical image, the user can find out where each event occurred on the C source program interactively.

Therefore, the user can recognize the accurate location where the largest memory area was used on source program and which memory allocation caused memory leak. It is efficient to optimize the usage memory size of C programs, because usually finding out the memory size on each line is very difficult. These functions are efficient for embedded system, whose memory size is limited strictly. The system just begins working and we could try some small test programs. These programs make and modify queue structure and binary tree structure. We have to evaluate the system performance by using more practical programs. More information is necessary for users to optimize the program.

For the future work, we would like to show the last access of dynamic allocated memory. This is the reason why it means the earliest point that the allocated memory can be release. We are considering using static data flow analysis and static type analysis. Tracking only uncertain reference as a result of static analysis, we can reduce log data and analyzing time.

References

[1] B. W. Kernighan and D. Ritchie, The C Programming Language (2nd Edition) , Prentice Hall, 1988.

[2] CoActionOS, [http://www.coactionos.com/embedded-design/101-understanding-memory-usage-in-c.html](Aug. 2013)

[3] [http://www.valgrind.org](Aug. 2013)

[4] A. Allain, "Using Valgrind to Find Memory Leaks and Onvalid Memory Use", [http://www.cprogramming.com/debugging/valgrind.html](Aug. 2013)

Run-Time Monitoring of Timing Constraints: A Survey of Methods and Tools

Nima Asadi, Mehrdad Saadatmand, Mikael Sjödin
Mälardalen Real-Time Research Centre (MRTC)
Mälardalen University, Västerås, Sweden
nai10001@student.mdh.se, {mehrdad.saadatmand, mikael.sjodin}@mdh.se

***Abstract*—Despite the availability of static analysis methods to achieve a correct-by-construction design for different systems in terms of timing behavior, violations of timing constraints can still occur at run-time due to different reasons. The aim of monitoring of system performance with respect to the timing constraints is to detect the violations of timing specifications, or to predict them based on the current system performance data. Considerable work has been dedicated to suggesting efficient performance monitoring approaches during the past years. This paper presents a survey and classification of those approaches in order to help researchers gain a better view over different methods and developments in monitoring of timing behavior of systems. Classifications of the mentioned approaches are given based on different items that are seen as important in developing a monitoring system, i.e., the use of additional hardware, the data collection approach, etc. Moreover, a description of how these different methods work is presented in this paper along with the advantages and downsides of each of them.**

Index Terms—Runtime Monitoring; Extra-Functional Properties; Real-Time; Timing; Survey.

I. Introduction

The number of computer systems used in our daily life and embedded as part of other systems, such as automobiles, microwave ovens, TV sets, etc., is exponentially growing. The interaction of such embedded systems with their surrounding environments (e.g., through sensors and actuators) often brings along timing requirements. Criticality of these timing requirements, of course, can vary from system to system and under different usage scenarios and situations. Therefore, ensuring that a system respects the timing requirements and operates within the timing constraints defined for it is of great importance and can even determine the success or failure of a computer system (e.g., the airbag system in a car).

While the goal of verification and validation techniques, such as testing, debugging, and theorem proving is to ensure general correctness of programs, the intention of run-time monitoring is to determine whether the current execution meets the specified technical requirements [1]. To achieve this goal, monitors collect the data of interest from the monitored systems, which can be used for further analysis by the user, or the monitor itself.

Timing behavior monitors provide the user with necessary information which can be used to detect or predict violations of timing constraints. Examples of such information are deadline misses and context switches. Many system performance monitoring tools have been developed. However, many of these monitors focus on different aspects of a system performance other than the timing behavior of the system, such as inter-process communications and/or access to shared memory resources. On the other hand, some of the methods used in such monitors are useful in timing performance data collection and analysis as well. Thus, a part of the effort in this paper has been dedicated to distinguishing and including those methods.

Various approaches were suggested in different areas, such as monitoring, debug and replay, and data analysis and visualization. However, the area covered in those studies has mostly been software-fault monitoring in general, i.e., monitors that are used to detect any sort of software fault. The focus in this paper is tried to be on approaches used for monitoring of timing constraints. The data that such monitors provide is especially very important for prediction and analysis of the performance of systems in real-time environment. Our goal is to provide the researchers and developers with a good insight to software monitoring approaches with a focus on timing constraints violation detection. To achieve this goal, an overview of the methods as well as an introduction to the used concepts and definitions is presented. The approaches covered in this study are tried to be a representative sample of timing constraints performance monitoring tools and relative studies.

The organization of this paper is as follows. Section 2 presents a background of the topic as well as the definitions of concepts used in this paper. Section 3 describes the methods used in selecting the monitoring approaches and a brief review of related work. Section 4 discusses the methods, the goal they try to achieve, and the advantages and disadvantages of each method. Section 5 summarizes the survey. The concluding remarks and the future work are given in Section 6.

II. Definitions and Background

Real-time systems are those systems in which the correctness of the system depends not only on the logical results of computations, but also on the time at which the results are produced [2]. Although significant work has been done to suggest a method that guarantees the execution of tasks within their pre-specified timing constraints, deadline violation can still happen due to different reasons, such as the unpredictability of the system environment and external signals, and the inability to satisfy all design requirements [3]. Software verification methods are used to make sure that the system meets its general requirements. However, despite the contributions of common verification methods and improvements in real-time scheduling, the need to perform run-time monitoring of these systems is not diminished due to the complexity of these systems and the unpredictability in dealing with the external environment [3]. Therefore, a monitoring tool can be helpful for detecting violations of those timing constraints by collecting, and analysing (depending on the facilities provided by the monitor) relevant system performance data.

According to Peters [4], a monitor is a tool that observes the behavior of a system and determines if it is consistent with a given specification. We decided to use Peters' definition of a monitor, because it covers different categories of monitoring systems. The system in which monitoring, run-time checking, or run-time verification is performed is referred as the 'target system', and the software application whose execution is being monitored is referred to as the 'target application'. The data detected by various monitors can be different. In this paper, our focus is on the data that can be used in analysis of timing behavior of different types of target systems, such as distributed systems, multiprocessor systems, and embedded systems, or information that can help detecting such data. Most of such monitors focus on detecting events of interest. An event is usually a state change in the target application.

1) Latency and interference: Event detection and processing can be performed in different ways, each of them causing a different amount of interference with the target system. The best solution for monitoring with respect to interference is a monitor that uses extra hardware to be able to detect events without affecting the activities of the monitored system. Such tools are usually referred as *hardware monitors*. Run-time monitoring without interference to the target system is usually accomplished by passively monitoring the target processor's data, address, and control buses [5]. *Passive monitoring* is a term used for a type of monitoring that does not affect the target system's performance. However, many state changes of the software being monitored are not reflected by *probes* that are created in data collection lines in the added monitoring hardware. Probes are basically elements of a monitoring system which are attached to the target system in order to collect information about its internal operation. If the internal state of the system context needs to be thoroughly known to make us able to detect an event, a monitoring tool which does not use extra hardware, called *software monitor*, is needed. However, implementing a software monitor needs modification on the target system kernel code, which can alter the behavior of it. To overcome this problem, a *hybrid monitoring* approach could be used. However, this approach that combines the hardware and software monitoring architecture suffers from the same limitations as the hardware monitoring approach. Besides that, the observations will be on a low amount of detail. In order to test and debug a system at satisfactory levels of reliability we need to observe the system completely. We can observe significantly more than it is possible with hardware monitoring approaches by including instrumentation code in the monitored software (application and kernel). Thus, for most application domains, pure software monitoring seems to be the better solution. This can be done by inserting small code stretches in the target program in order to detect events of interest. Different from hardware monitoring systems, software monitoring systems are easier to change. Besides that, the flexibility (modifiability) of software monitoring approach makes it possible to provide more information to programmers and in general, to provide information in a more useful form [5].

As mentioned, including the monitoring code in the target software has the disadvantage of changing its behavior because of the amount of latency being added to it. This is because a part of the CPU time should be dedicated to the monitoring code. This latency is referred as probe effect.

As for the use of the monitoring results, monitors have to choose between a low latency and a small rate of evaluations. Because evaluating a big amount of collected data can increase the latency in the system. The amount of latency usually depends on the focus of monitoring tool and methods. Monitors that only gather data for later use can usually cope with a large latency, whereas monitors that control the monitored system based on the results of evaluations will require a low latency [6].

2) Tracing and Sampling: Data collection can happen in two ways: *tracing* and *sampling*. In tracing, every occurrence of an event creates a record. So event tracing is characterized by the completeness of knowledge [7]. Sampling yields only a statistical measure of the software's execution patterns. It is not precise: if an event does not occur in a sampling log, there is no guarantee that it did not occur in execution. This means that sampling may not be able to detect frequently executed routines whose execution times are smaller than the sampling frequency. However, significantly less time needs to be spent to achieve sampling than to instrument the software system for tracing [7]. Also, the data volume associated with event tracing can be very large. Regarding interference and target behavior change, both event tracing and sampling may affect the performance of the software system. In some literature, tracing is mentioned as *event-driven monitoring* whereas sampling is called *time-driven monitoring*.

III. Survey Methodology

Many run-time monitoring methods have been developed in the past. These methods serve different goals by gathering data of interest from different aspects of systems. Thus, the effort of this paper is to provide an informative categorization of the monitoring tools based on these differences. In this section, the motivation behind this survey and the methods used for the survey are discussed.

A. Objectives of this survey

With ever increasing use of real-time systems, the reliability of such systems seems more and more crucial. In order to make sure that the real-time characteristic of the system is preserved, many techniques for run-time monitoring and debugging of these systems have been developed. In general, monitoring supports the debugging, testing, and performance evaluation of the programs, and covers different aspects of system's behavior, such as memory usage, CPU usage, network and connections status, and tracing of the execution of the processes in the system. Monitoring of timing constraints focuses on the timing behavior of the processes in the system. The main goal in monitoring in this area is to make sure that the real-time quality of the system is guaranteed, which basically means that all the tasks are completed without missing their deadlines. In order to have a solid vision of the performance of a system regarding this quality, it is important that the monitor can provide the data needed for the analysis of timing behavior of the system. The goal of this work is to present an overview of the architecture and workflow of monitors which can be used for timing analysis. A few surveys have been done on run-time monitoring. Moreover, some of the existing works, such as the work of Delgado et al. [1], have tried to cover a wider range of run-time monitors. We tried to narrow down our survey to the monitors whose data can be used in timing analysis of systems. Also, many of the methods covered in this work are not covered in the work of Delgado. The approaches covered in this work are representative samples of such monitoring tools. The categorization that is brought in the summary section is based on the design and architecture of the monitors, the services they provide for the user, and their other important features. Furthermore, the positive and negative points of each method are presented along with the explanation of them to help the readers gain a better vision about them.

B. Related Work

A close work to our work is the taxonomy of run-time software fault monitoring by Delgado et al. [1]. In that work, a classification of tools that monitor software faults is presented. Reinhard Wilhelm et al. [8] discuss the issues in Worst-Case Execution Time (WCET) analysis and review the common suggested tools for this purpose. They divided the tools into two main categories: static methods and measurement based methods. Another survey related to monitoring of system performance is the work of Henrik Thane [9]. Besides an explanation of common concepts and terminologies in performance monitoring, he provides a short review of some of the suggested monitoring methods. In that work, monitors are classified as hardware monitors, software monitors, and hybrid monitors, which are a combination of the first two. A bibliography of the works on performance evaluation was presented by Agajanian in 1975 [10]. Gu et al. provide a review on the literature on monitoring and debugging in their annotated bibliography [11]. They divide their work into four section including modeling and design of the systems, data collection, analysis of the collected data, and dynamic performance controlling. Also, a number of bibliographies of parallel debugging tools were presented by Pancake et al. [12] [13] [14].

C. Review Method

Certain literature review guidelines and approaches were taken into consideration to choose the papers that cover our topic of interest. The application of those approaches is only briefly explained in this section due to space limitations.

a) Inclusion and exclusion criteria: Studies that presented data about software monitoring or performance evaluation were included in the paper data base. The outcome of the studies was not considered in the inclusion criteria. Papers that were published up to 2012 were included in this survey.

b) Search Strategy: The main resources we used to access the papers of interest include the following: ACM Digital Library, IEEE Xplore, ScienceDirect Elsevier, SpringerLink. The main keywords that we used for searching include: Monitoring AND run-time AND software, Performance evaluation AND run -time AND software, Performance Evaluation AND real time AND WCET, Analysis AND run-time AND software, Analysis AND Linux AND run -time, Analysis AND timing constraints. Apart from these main ones, OR combination of some of these keywords were tried and executed as well.

c) Using Citation for Inclusion: Finding the papers that cited a specific paper was the first step of this strategy. Among the papers that were found this way, a number of them were selected according to their relevance to our topic of interest. Specifically, the papers that were about monitoring of irrelevant systems were removed from our survey. For indicating if a paper was relevant or not the whole paper was skimmed or read, because the abstracts would not always give information on whether the paper presented empirical results or not. Another strategy in citation management was to search for the papers cited in the related work section of studied papers. The same relevance check criteria went on for those studies as well.

IV. Overview of the methods

This part presents an overview of the architecture and design of the suggested approaches for the monitoring of system performance regarding to timing constraints. The aspect of monitoring that each work aims to improve or resolve is also stated. Moreover, some of the Pros and Cons of each suggested solution are presented at the end of each section.

A. Non-interference method

1) Objective: to provide a monitoring and debugging system that ensures minimum intervention with the execution of the target system.

2) Approach: The monitor architecture consists of two main parts: the interface module, and the development module [15]. The interface module's major duty is to latch the internal states of the target system based on predefined conditions set by the user. The responsibility of the development module, which contains a general purpose microprocessor, is to start the monitoring process, to record the target node execution history, and to perform analysis on the recorded data. After being connected to a node of the target system and initialized, the interface module keeps collecting events of interest until it finds a stop condition, pre-specified by the user. Then an interrupt is sent to the monitoring processor to separate it from the target processor for the data recording process to take place. The recorded information is transferred to a secondary storage for further processing. The events of interest include process-level events. During the monitoring, the time at which each event happens is recorded. Using this timing information, the execution history can be examined against timing constraint requirements. If violations are found, the replay mechanism can be used to test the program behavior again in order to isolate the errors.

3) Advantages: The monitor imposes low interference to the target system. Also, the start and stop conditions can be planned by the user, which makes this method more flexible.

4) Disadvantages: Generating an interrupt for every event occurrence imposes unpredictable interference to the target system. However, this is the only interference of the monitor with the target system. There is no guarantee that the microprocessor buses of the future will have the properties required to support bus snooping [16], a technique to achieve cash coherence in distributed systems that this type of monitors rely on.

B. PASM

1) Objective: Suggesting a monitor with a flexible specification language to provide the user with automatically defined process-level events to associate them with actions to be taken by a hardware monitoring system.

2) Approach: PASM citeLumpp1 [17] is a programmable hardware monitor, which provides a flexible set of tools for the user to specify events for a wide variety of monitoring applications. The user can include a monitoring section with the application that defines events of interest, actions to be executed upon detection of those events, and the binding of events to actions. This section is then used by the compiler to automatically implement the instrumentation. Events in this monitor are associated with changes of state of the active process. An action can be recording the time of the occurrence of an event to track the timing behavior, or printing of the contents of some internal data structures when a certain point in the execution is reached.

3) Advantages: The programmer has the freedom to define many types of events as functions of the monitored data, and actions corresponding to them. The monitor imposes low interference with the target system. Manual instrumentation is however hard, time-consuming and prone to error, so the automatic instrumentation suggested in this approach removes this problem.

4) Disadvantages: parts of the code, which were affected by the monitoring sections, need to be recompiled when the programmer wants to modify the probes.

C. ART Real-Time Monitor

1) Objective: The objective of ART Real-Time Monitor is to visualize the system's internal behavior with lowest amount of change in its timing behavior.

2) Approach: This monitoring system was developed for ARTS, a distributed operating system developed in 1980 [18] [19] [20]. This approach focuses on visualizing the timing behavior of the system processes. Rate monotonic and deferrable server algorithms are supported by this monitor, and the monitoring task is performed as a part of the target system. The functional structure of the monitoring system can be divided in to three major parts: a part of the target operational system code that records the information of interest about the processes, called event trap, the reporter, which sends the information to the visualizer, and the visualizer, that uses the resources sent from the target system to create historical diagrams of the scheduling decisions of the target system.

An event is generated each time the state of a process is changed. The ARTS monitor records process-level events such as process-creating, waking-up, blocking, scheduling, freezing, killing with completion, killing with missed deadline, and killing with frame overrun [18]. For monitoring timing constraints, the monitor uses the facilities that the ARTS kernel provides, such as 'Time fence'. The 'time fence' is a mechanism in the ARTS Kernel used to detect a timing error at run-time. Before each operation invocation the time fence is checked to verify that the slack time is bigger than the worst case execution time of the invoked operation, and a timer is set. If the execution is not completed within the worst case time, the timer announces an anomaly.

3) Advantages: The integrated scheduler uses rate monotonic scheduling for periodic hard real-time tasks and deferrable server for aperiodic soft real-time tasks. Also, separation between the reporter and the Visualizer makes the monitor suitable for embedded systems.

4) Disadvantages: interactive debugging of real-time systems without deterministic replay is not enough for removing errors because debugging commands can damage the timing-dependent nature of real-time systems [21]. Also, the monitor needs extra kernel support from ARTS, which makes it invasive. If the target system does not provide sufficient resources, the monitoring capability will be limited, consequently, thus, a hybrid approach which uses extra hardware might be necessary.

D. Hmon

1) Objective: To design a transparent monitoring system with continuous data collection facility for HARTS distributed system.

2) Approach: Hmon [21] was developed to monitor the performance of the Hexagonal Architecture for Real-Time Systems (HARTS), a distributed real-time system. The area this monitor covers include monitoring interrupts and shared memory as well as the calls that the users can use in order to monitor the processes that are not covered by the monitor. The monitoring is done by including the monitoring code in the existing system call libraries, meaning that no inside kernel changes are necessary. Context switch events are detected via a hook provided by the pSOS kernel. Task scheduling and CPU usage are determined by studying the order and timing of these events. Also, process management calls, such as process creation and deletion, and time management calls that set or read the clock are recorded to obtain their real-time properties.

3) Advantages: the monitoring is performed transparently, so the programmer does not need to add special code to applications. Also, some system hardware is dedicated to the monitor to minimize interference with the measured system, but no special hardware is required. The system is intended for general-purpose real-time multiprocessors.

4) Disadvantages: data collection code interferes with the system being monitored, and can change the system behavior.

E. Halsall-Hui

1) Objective: to design an interactive monitoring tool for system and application level monitoring that is suitable for embedded systems.

2) Approach: This monitor is designed to gather the data from each processing node of a real-time embedded system which is based on a distributed architecture [22]. The recorded information from each node includes the IDs of the tasks and processes, the type of the tasks and the system calls, and the time that those events happen [22]. The event data recording can take place in two ways. In the first method, the user inserts a library function call, with the corresponding variable name as a parameter, at the appropriate point in the source code, so that whenever that code section is being executed the recording function is called and run. Each of these functions can record a specified event. The event data is then sent to the single monitor of that node, which is also a library function. In the second method, an interrupt is used to periodically refer to a data table provided by the user, which includes event identities, the recording frequency, and the variables to be recorded. This event information is saved in a system file, or an application file, which is downloaded to the monitored system later.

3) Advantages: The method allows application-specific events to be monitored and analysed. The monitor is modifiable, and the monitor interference does not change during replay, which makes the timing behavior of the target system more predictable

4) Disadvantages: This method is invasive and not appropriate real-time systems because of its high amount of interference.

F. Hybrid Monitor

1) Objective: To design a monitor that combines the flexibility of software methods and non-interference of the hardware methods.

2) Approach: This monitor is designed by combining hardware monitors and software monitors [23]. A Test and Measurement Processor (TMP) is integrated to each node of the distributed system in order to record their process and intercommunication activities. The main principle of this method is that the target system generates events of interest, and the TMP hardware processes and time stamps them [23]. The collected event data is stored in a FIFO memory in the CPU. Every time an event is sent into the FIFO buffer the CPU of the TMP is notified by an interrupt activating the processing of the events data. The number of messages, the message length, failed messages, the system time (the time spent for the processes in kernel), and the application time (the time that application processes spend in kernel minus the system time) can be measured using this monitor as well. Events are time stamped locally in this method.

3) Advantages: This method is transparent, i.e., it does not change the behavior of the system, thus the monitoring is continuous. It also uses hardware support to have a low overhead for typical applications. Also, the graphical representation helps better understanding of the recorded data. Furthermore, since the TMPs communicate via their own network, the communication disturbance to the host system is lowered. Another positive point about this tool is that users can load their own evaluation software instead of the default TMP analysis software.

4) Disadvantages: Obtaining hybrid schemes is generally hard. One reason is a lack of architectural support for the monitoring hardware. Standard interfaces are needed to generate industry participation and allow instrumentation portability [24]. Although the analysis part can be changed, till this tool is not flexible for manipulation by the user(for example event detection and type of data being recorded are not decidable). The overhead, although claimed to be small according to the implementation for typical applications (0.1%), is not negligible for real time applications.

G. ZM4

1) Objective: To develop a hardware event driven monitoring tool for parallel and distributed systems.

2) Approach: In this approach, a hardware system called ZM4 , and an event trace processing software called SIMPLE, which works independently from the monitor, are developed [25]. The connection between the hardware and the monitored system is local area network type. Hosting the monitoring system, storing the measured data, and presenting an analysis interface for the users are the responsibilities of the control and evaluation center (CEC) of the ZM4 system. Also, a number of monitor agents are built as slaves for the CEC. Each monitor agent is connected to a target system node. Another responsibility of these probes is time stamping and recording of the events. Time stamping is done using a global clock with the precision of 100 ns. SIMPLE, which works on Linux and MS-DOS, is an evaluation environment used for analyzing the recorded event traces. It generates a global view of the distributed system's behavior and performs trace validation and analysis as well. Whenever the monitor recognizes an event, it stores an event record which consists of event token and a time stamp. The sequence of events is stored as an event Truce [25].

3) Advantages: Distributed hardware monitor ZM4 can be adapted to arbitrary target systems. The combination of event-driven monitoring and event-based modeling makes program instrumentation and validation systematic.

4) Disadvantages: Reliance on event driven monitoring and instrumentation is limited to limit the impact on the target system.

H. Trams

1) Objective: To design a hybrid monitoring method for the performance measurement in both tightly and loosely coupled multiprocessors.

2) Approach: The architecture of this system consists of a software for event triggering, by inserting Write command in the code, and a hardware subsystem used to sample the time and identity of the CPU [26].

For the hardware part, a measurement node consists of a set of VLSI chips with two IC chip types: the Trams (Trace Measurement System) and the Rems (Resource Measurement System). The data written by the user, along with the CPU identification and the time stamp are stored in the Trams sample memory. The sample memory then reads this information for further analysis. Rems is used for data sampling. The target distributed system can be tightly coupled or loosely coupled. In the first system a single node can be used in a centralized event trace collection, and in the second architecture model each node can be connected to a corresponding processor. Both the Trams and Rems contain three sections: a data capture system, an output, and a FIFO buffer.

3) Advantages: Both loosely coupled and tightly coupled systems are covered in this approach. As a special feature, event counters are implemented in one of the VLSI chips in order to reduce the amount of data to be transferred and evaluated [26].

4) Disadvantages: The monitoring tool is system specific, and it makes it easy to generate so much data that it swamps any file system or data analysis station [27].

I. Alamo

1) Objective: A method to reduce development costs for a broad class of execution monitors.

2) Approach: Lightweight Architecture for Monitoring (Alamo) [28] [29] [30] is an event-driven monitor developed for C programs, and uses the Icon programming language to specify assertions. The Alamo monitoring architecture utilizes CCI, a Configurable C Instrumentation tool as a preprocessor that uses parse trees to identify monitoring points and inserts events into the target program source code. The architecture of this monitor consists of: (1) an automatic instrumentation mechanism, (2) an execution model, (3) abstractions for event, selection, multiplexing and composition, and (4) an access library that allows monitors to directly manipulate target program state. Alamo employs automatic program instrumentation to produce target program events for the monitor. The Execution Monitor (EM) executes the Target Program (TP) and then returns control with information in the form of an event report. The user can apply a predicate to each event report to make monitoring more specific, or view detailed information through Alamo's visualization mechanism.

3) Advantages: The Alamo monitor architecture significantly reduces the development cost of writing program execution monitors

4) Disadvantages: There is no support for real-time or shared-memory multiprocessor-based parallel applications. Not all execution monitors can be written using an Alamo-based framework; those that, cannot tolerate intrusion of instrumentation code require a two-process model such as that employed by standard source-level debuggers [31].

J. MAC

1) Objective: To propose a tool that complements testing (infeasible to completely test the entire system due to the large number of possible behaviors), and verification (possibilities for introduction of errors into an implementation of a design that has been verified) techniques.

2) Approach: Monitoring and Checking (MAC) [32], [33], [34], [35], [36], [37] provides a framework for runtime monitoring of real-time systems written in Java. The MAC architecture consists of three main components: a filter, and event recognizer, and a run-time checker. The filter, which maintains a table containing names of monitored variables and addresses, extracts low-level information, time stamps it, puts it in a message, and sends it to the event recognizer. From this low level data, the event recognizer detects the occurrence of abstract requirement level events based on the Requirement specifications written in Meta Event Definition Language (MEDL), and informs the run-time checker about them. The run-time checker uses these events to see if the current system execution conforms with the requirements of the system. An event is an instantaneous state change. Static analysis is used to determine monitoring points, which are inserted automatically.

3) Advantages: The filter (that extracts the information of interest) is separated from the event recognizer, so that system execution does not suffer from the overhead of abstracting out events from low-level information. This architecture is also appropriate for monitoring distributed systems where each module is able to have a corresponding filter.

4) Disadvantages: This architecture adds to the communication overhead because the filter sends the data to the event recognizer. the executing software needs to send enough state information to observer process, in order to check constraints and do analysis. When violation of the constraints happens, observer process cannot stop the execution of the software.(there is no feedback to the system), but this is feature is added in MACS, a later work [33].

K. PMMS

1) Objective: To minimize the total time between formulation of the questions (what the monitor should do) and delivery of the answers. The second is to minimize the monitoring overhead during execution.

2) Approach: Program Monitoring and Measuring System (PMMS) [38]. is a monitoring approach that automatically collects high level information about the execution characteristics of a program. Data collection is done by code inserted into the source program of the target system, and conditions are used to filter out events that are not relevant. The monitor handles events of interest by installing code that reacts whenever they occur. This data collection code includes Pre-condition (relevance test), Local-variables (used to store local data), Before-code (code to collect data available before the event), After-code (code to collect data after the event), Post-condition(a relevance test based on data that is available after the event), and Action(code that stores data more permanently for later use) [38]. Examples of recorded event data include the time at which the event occurs, the value of program variables at that time, etc. The user can specify all the objects and relations using the high level specification language that is provided in this method. The PMMS uses a main memory for the active database to facilitate the collection, computation, and access to the computation results.

3) Advantages: The used specification language in this work allow security engineers to write a centralized policy specification; the systems then uses a tool to automatically insert code into untrusted target applications. This centralized policy architecture makes reasoning about policies a simpler and more modular task than the alternative approach of scattering security checks throughout application or execution-environment code. With a centralized policy, it is easy to locate the policy-relevant code and analyze or update it in isolation [39].

4) Disadvantages: Since the instrumentation code performs database queries, instrumentation can significantly change the performance of the target program.

L. JRTM

1) Objective: An approach for monitoring timing constraint violations in real-time systems. The objective is to detect timing violations as early as possible.

2) Approach: Java Runtime Timing-constraint Monitor [40], [41] targets timing properties of distributed, real-time systems written in Java. In this work, the necessary constraints and event log are automatically derived by the compiler, and then the compiled specification is loaded into the monitor at run-time. Java programmers can insert the event triggering method calls in their Java programs where event instances are supposed to occur. At run-time, whenever an event method is executed, the current system time is recorded as the event occurrence time and this timestamp is sent to the monitor along with the event name. The monitor keeps these event occurrence messages in a sorted queue with the earliest event message at the head of the queue. The event message at the head is processed at an appropriate time to check it with the related constraints. Once a violation of the specification is found, users are notified. This monitor can run on the same machine as the target process or on a standalone monitoring machine.

3) Advantages: Low overhead; it uses small size of event record history depending on the maximum occurrence rate of events.

4) Disadvantages: It is difficult to timestamp an event with an accurate time point, which is assumed to be measured well for JRTM to use.

M. GRTMon

1) Objective: To design a run-time monitor with small probe effect, and no input missing (not for non-real-time purposes).

2) Approach: Generalized Run-Time Monitor (GRT-Mon) [42] is a tool for real-time systems to detect information regarding timing constraints. In this method, data collected by sensors is written to buffers from which monitors read. Each buffer is mapped at the respective sensor section and all associated monitor tasks. According to the work flow of this monitor, data pairs of an output element and its timestamp are the input to evaluation algorithms of the monitor. Monitors sort the buffer output elements based on their timestamps before evaluation. The CPU's timestamp counter, which contains the number of elapsed CPU cycles since the CPU has been initialized is used by the sensor to tag the output with its corresponding timestamp. A sensor directory is used to provide relations between sensors and monitors. Thus, there is no direct relation between sensors and the monitor, which can be effective in decreasing the probe effect of the monitor. The monitor can either run as a constant-bandwidth server with a bandwidth that the user defines, or resource requirements can be determined based on the sensors' jitter-constrained stream specifications [42]. Also, in GRTMon, monitors and the target system communicate asynchronously, so the monitors have less direct influence on the monitored system's timing. Examples of events of interest are context switches, inter-process communication (IPC) or events in the kernel itself such as calls to certain kernel functions.

3) Advantages: Using this method evaluation of events with least amount of input data miss is guaranteed. Also, small set of dependencies between the monitor and the target system and sensors and the monitor decreases the overhead on the target system.

4) Disadvantages: If more than one sensor is used the overhead will increase significantly.

N. FKT

1) Objective: To design a simple software monitor for Linux with lower interference which can support multiprocessor platform and networked environment.

2) Approach: Fast Kernel Tracing (FKT) [43] monitor is a software tool designed to evaluate the performance of Linux kernels running on Pentium PCs. This monitor is implemented by modifying the Linux kernel through adding probes for data collection, and user-level programs for data evaluation. The probes are placed by the programmers. By default probes are placed at the entry to and exit from every system call, trap, interrupt, and process switch inside the kernel [43]. The timing recorded by a probe is the time provided by the Intel Pentium's timestamp counter which is incremented on every hardware clock cycle. The data recorded by the probe consists of the time at which the data is recorded by the probe, a unique identification code assigned to the probe, the ID of the current process, the number of the processor, and additional parameters provided by the programmer. The monitor has two phases: recording, which happens during the run-time, and analysis, which happens off-line. The analysis part can be changed by the user for different types of evaluations. Also, the information to be collected can be specified by the programmer while inserting the probes.

3) Advantages: The probes can be turned on and off using a key mask that is controlled by user-level programs, so that the probing overhead is reduced when probes are not needed to be used. Also, the amount of information recorded by each probe is small, which means that big traces of operating system execution can be recorded.

4) Disadvantages: This tool does not provide a run-time analysis of data, so the user does not notice violation occurrence during the run-time. When the buffer is filled the probing is suspended, which implies the use of a big buffer.

O. SoC-based Monitor

1) Objective: A runtime monitor within an embedded system to detect timing specification violations

2) Approach: The System on Chip-based monitor [44] uses a hybrid method for run-time verification of embedded systems. The monitor consists of event recognizer, a verification tool, and the monitor output. The event recognizer decides if the collected data is relevant to the event definition. After passing this step, the event data is sent to verification section where it is compared with the requirement constraints. In case a violation is observed, it is sent to the output of the monitor. The events detection code is inserted in the source code of the target system, but no code is needed for transmitting the events to the event recognizer. In fact, the event data is transmitted from the target system to the event recognizer by a dedicated monitoring core called 'event dispatcher' [44].

3) Advantages: Low overhead due to use of extra hardware for event dispatching. It benefits from a light design for monitoring of embedded systems.

4) Disadvantages: Limited monitoring is available due to the memory constraints of embedded systems. The monitor's performance is highly dependent to the target system hardware specifications.

P. Raju-Jahanian

1) Objective: Early detection of violations of timing assertions in an environment in which the real-time tasks run on multiple processors

2) Approach: This monitoring tool consists of a set of cooperating monitor processes one on each processor of the target system [3]. Upon occurrence of an event, application tasks on a processor inform the local monitor by putting the event into a queue in shared memory. Then, a monitor process decides whether the event must be communicated to other monitors or not. The role of this monitor is to make sure the violation is predicted as early as possible [45], by deciding if the data is communicable or not using intermediate constraints. The main idea behind this solution is that 'it is possible that an implicit constraint is violated before an explicit delay or deadline becomes unsatisfiable at run-time' [45]. If the occurrence time of an event has to be sent to a remote monitor, the monitor puts the event and its local occurrence time into a message and sends it to other monitor processes. If a message arrives from a remote monitor or a timeout occurs, a monitor checks if violation has occurred. If a violation is detected, it notifies the application task (with termination as the default action).

3) Advantages: The intermediate monitor makes early violation detection possible

4) Disadvantages: It uses Real-Time Logic specification language (RTL) for constraints and event-action bonding, which is rarely used in practice.

Q. OSE Monitor

1) Objective: To facilitate the possibility of monitoring of timing behavior for OSE real time operating system.

2) Approach: The main idea behind this approach is to add a second layer scheduler to the OSE (Operating System Embedded) real-time operating system to make it easier to query the execution result of real-time tasks [46]. This adjunct scheduler uses the specifications of real-time tasks, such as the period and execution time of each task, from a parameter file. According to these parameters the second layer scheduler schedules the tasks by allowing them to be sent to the core scheduler in Earliest Deadline First(EDF) or Rate Monotonic Scheduling (RMS) scheduling algorithms. Thus, it is clear that the second layer scheduler process must have the highest priority among all the OSE processes.

The monitor process works with the lowest priority, i.e., as a background OSE process, in order to make sure that it does not interfere with the scheduling process. Upon completion of a task, the monitor receives a signal from the second layer scheduler. Two types of log files are created in this process: a scheduling log file, and a monitoring log file. The scheduling log file, which is created by the second layer scheduler, contains the time points at which a task in the task set is scheduled, completed, or preempted. Monitoring log file, which is created by the monitor, is updated only when an instance of a task is completed [46].

3) Advantages: A very good set of timing information is provided by the log files without further analysis processes, which makes this tool very easy to use.

4) Disadvantages: Dynamic creation of tasks is not covered in this method. The overhead of another scheduling layer on the real-time system can be significant.

V. SUMMARY

A number of suggested tools were selected out of a bigger group of studies on system monitoring and performance evaluation. As mentioned before, the focus of this paper is on the tools and methods whose presented data can be used for timing analysis of the system performance. Other

TABLE I
A CLASSIFICATION OF MONITORS

Approach	Monitor Adaptibility		Data Collection		Design Method			Development stage		Target System		
	Specific	General	Tracing	Sampling	Hardware	Software	Hybrid	Research	Production	Real-Time	Embedded	Distributed
Non-inter	x			x	x			x		x		x
PASM	x		x		x			x				x
ART			x		x			x		x		x
HMON	x			x		x		x		x		x
Halsall-Hui		x	x	x		x		x		x	x	x
Hybrid		x	x				x	x				x
ZM4		x	x				x	x				x
Trams		x	x				x	x				x
Alamo	x		x			x			x			
MAC	x		x			x			x	x		
Pmms		x	x			x		x				
JRTM	x		x			x		x				x
GRTMon		x	x			x		x		x		
FKT	x		x			x		x				
Soc-based		x	x				x	x		x	x	
Raju-Jahanian		x	x			x		x		x		x
OSE monitor		x	x			x		x		x		

performance evaluation approaches such as debugging, testing, and visualizing were not covered in this survey.

In this section, a classification of the reviewed tools is provided in Table I. This classification is based on the features that can be useful in giving the developers and researchers a broad insight on different suggested approaches in designing system performance monitors. These features were chosen in order to satisfy the goal of facilitating the process of research on run-time monitoring of timing properties for the readers. A description of each classification element is provided in the sections below

A. Monitor Adaptability

Depending on the design purpose, some of the monitoring tools are developed for a specific target system. In many cases, the architecture of such monitors is dependent to the facilities that the target system provides. Monitors that are not designed for a specific target system can provide the developers the possibility of designing transparent monitoring for target systems with basic facilities and source code in any programming language. In our classification, 'General' adaptability means that the monitoring method can be used for different types of target systems. We chose the term 'specific' for the tools that were developed for a specific target system, or monitor programs in a specific programming language, and is not not possible to be implemented for other systems.

B. Data Collection Method

An important task of any a run-time monitor is to collect the data of interest from the monitored system when it is running. Two types of data collection during the system execution are sampling and tracing. A brief description of the two mentioned methods was previously given.

C. Design Method

As explained in the prior sections, depending on the use of extra hardware in the monitoring system, a monitor can be hardware, software, or a combination of the two, called hybrid. A description of the advantages and drawbacks of each type is given in the previous sections.

D. Development Stage

While some of the covered methods were employed in software production projects, thus are available tools, the others are classified as research project prototype.

E. Target System

As mentioned in previous sections, the monitors covered in this work are designed for different environments and platforms of target systems ranging from embedded systems to distributed and parallel systems. This section on the table represents the type of target systems that the monitors were designed for, or can be used for. Some monitors, such as FKT, were designed for general-purpose systems.

VI. CONCLUSION

There is an increasing need in monitoring of timing behavior in different types of computer systems. This is mainly because of the growing importance of the issue of satisfying timing constraints in many systems that are being used today, particularly embedded devices. A practical and reasonable method for controlling a system's timing behavior is through run-time monitoring of timing in the system. In this paper, we provided a survey of a selected group of works on monitoring of timing constraints in different systems and contexts. The systems in need of monitoring covered in this work ranged from embedded systems to hard real-time and distributed systems. Our main intention with this work has been more to gather versatile monitoring contexts and methods than merely analyzing monitoring methods targeted for a single specific context or monitoring methods using the same design architecture (both in terms of hardware or software implementation). For each approach that was covered, a review of its work flow and design of each was presented as well as their advantages, drawbacks, and the problem each of them aim for. Then, a short summary and a classification of the methods were offered based on the each method's architecture and other practical features.

Software and hardware monitors have been developed to tackle different monitoring needs and to enable collection of data considering the interference of the monitor in the target system's performance, which is referred to as probe effect. In this sense, hardware monitors try to minimize the interference and performance penalty of monitoring, while software monitors generally provide a more flexible and customizable solution. Also, hybrid monitors have been designed as a combination of the two mentioned architectures in order to resolve their issues, and benefit from the advantages of each. However, due to the complicated nature of timing behavior of systems, and the increasing complexity of different systems, adaptation and customization of existing methods may be required to match the needs of different systems and contexts. Hence, this paper's effort in summary has been on giving system designers and developers an organized insight toward the important available experiences in this area. This is achieved by not only describing different monitoring methods for different contexts, but also providing a classification framework for them.

VII. Acknowledgement

This work has been partially supported by the Swedish Knowledge Foundation (KKS) through the ITS-EASY industrial research school [47], and also by Combitech [48] and Xdin [49] companies in Sweden.

References

[1] N. Delgado, A. Q. Gates, and S. Roach, "A taxonomy and catalog of runtime software-fault monitoring tools," *IEEE Trans. Software Eng.*, pp. 859–872, December 2004.

[2] J. A. Stankovic and R. K., "What is predictability for realtime systems," *Springer*, November 1990.

[3] S. C. V. Raju and F. Jahanian, "Monitoring timing constraints in distributed real-time systems," in *Proc. Real-Time Systems Symp.*, vol. 30, pp. 57–67, 1992.

[4] D. K. Peters and D. L. Parnas, "Requirements-based monitors for real-time systems," *ISSTA '00, ACM Press.*, December 2002.

[5] S. Ricardo and J. R. De Almeida, "Run-time monitoring for dependable systems: an approach and a case study," in *in Proceedings of the 23rd IEEE International Symposium on Reliable Distributed Systems (SRDS 2004)*, vol. 30, pp. 41–49, October 2004.

[6] T. Riegel, "A generalized approach to runtime monitoring for real-time systems," *Master's thesis, TU Dresden*, 2005.

[7] E. Metz, R. Lencevicius, and T. F. Gonzalez, "Performance data collection using a hybrid approach," in *Proceedings of the 10th European software engineering conference held jointly with 13th ACMSIGSOFT international symposium on Foundations of software engineering*, vol. 30, pp. 126–135, 2005.

[8] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckmann, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenstrom, "The worst case executiontime problem, overview of methods and survey of tools," *Trans. on Embedded Computing Sys.*, 2008.

[9] H. Thane, "Testing and debugging of distributed realtime systems," *PhD Thesis, Mechatronics Laboratory, Royal Institute of Technology, Stockholm, Sweden*, May.

[10] A. H. Agajanian, "A bibliography on system performacne evaluation," *Computer*, November 2000.

[11] W. Gu, J. Vetter, and K. Schwan, "An annotated bibliography of interactive program steering," *ACM SIGPLAN Notices*, September 1994.

[12] S. Utter, C. M. Pancake, and K. Schwan, "A bibliographyof parallel debuggers," *ACMSIGP1un Notices*, pp. 29–42, November 1989.

[13] C. M. Pancake and S. Utter, "A bibliographyof parallel debuggers," *ACMSIGP1un Notices*, pp. 21–37, January 1991.

[14] C. M. Pancake and R. H. B. Netzer, "A bibliographyof parallel debuggers," in *Proc. of the 3rd ACM/ONR Workshop on Parallel and Distributed Debugging, San Diego, CA, USA*, May 1993.

[15] J. J. P. Tsai, K. Y. Fang, and H. Y. Chen, "A noninvasive architecture to monitor real-time distributed systems," *Computer*, pp. 11–23, March 1990.

[16] M. M. Gorlick, "The flight recorder: An architectural aid for system monitoring," in *Proc. ACM/ONR Workshop Parallel and Distributed Debugging*, pp. 175–183, 1991.

[17] H. J. Siegel, T. Schwederski, J. T. Kuehn, and N. J. Davis, "An overview of the pasm parallel processing system," *in Tutorial, Computer Architecture*, pp. 387–407, 1987.

[18] H. Tokuda, M. Kotera, and C. W. Mercer, "A real-time monitor for a distributed real-time operating system," in *Proceedings of ACM SIGOPS and SIGPLAN workshop on parallel and distributed debugging*, May 1988.

[19] H. Tokuda and M. Kotera, "A real-time tool set for the arts kernel," in *Proceedings of 9th IEEE Real-Time Systems Symposium*, December 1988.

[20] H. Tokuda, M. Kotera, and C. W. Mercer, "An integrated time-driven scheduler for the arts kernel," in *Proceedings of 8th IEEE Phoenix Conference on Computers and Communications*, March 1989.

[21] P. S. Dodd and C. V. Ravishankar, "Monitoring and debugging distributed real-time programs," *Software Practice and Experience*, pp. 863–877, October 1992.

[22] F. Hasall and S. C. Hui, "Performance monitoring and evaluation of large embedded systems," *Software Engineering Journal*, pp. 184–192, 1987.

[23] D. Haban and D. Wybranietz, "A hybrid monitor for behavior and performance analysis of distributed systems," *Software Engineering Journal, IEEE Trans. Software Eng.*, pp. 197–211, February 1990.

[24] C. Alexander, "Multicomputer performance monitoring: a standards-based approach," *Technical Report MSSU-EIRS-ERC-93-13 Mississippi State University*, December 1993.

[25] R. Hofmann, R. Kar, B. Mohr, A. Quick, and S. M., "Distributed performance monitoring: Methods, tools, and applications," *IEEE Trans.Parallel and Distributed Systems*, 1994.

[26] A. Mink, R. Carpenter, G. Nacht, and J. Roberts, "Multiprocessor performance-measurement instrumentation," *Computer*, pp. 63–75, September 1990.

[27] J. K. Hollingsworth, B. P. Miller, and J. Cargille, "Dynamic program instrumentation for scalable performance," in *Proc. Scalable High-Performance Computing Conference, Knoxville, Tenn.*, pp. 841–850, 1994.

[28] C. L. Jeffery, "Program monitoring and visualization: An exploratory approach," *Springer-Verlag*, 1999.

[29] C. L. Jeffery *New Mexico State Univ., Las Cruces, N.M., personal comm.*, 2002.

[30] C. L. Jeffery, W. Zhou, K. Templer, and M. Brazell, "A lightweight architecture for program execution monitoring," in *Proc. ACM SIGPLAN/SIGSOFT Workshop Program Analysis for Software Tools and Eng.*, pp. 67–74, 1998.

[31] C. L. Jeffery, "The alamo execution monitor architecture," *Electronic Notes in Theoretical Computer Science*, 2000.

[32] M. Kim, S. Kannan, I. Lee, O. Sokolsky, and M. Viswanathan, "Javamac: A run-time assurance tool for java programs," in *Proc. Fourth IEEE Intl. High Assurance Systems Eng. Symp.*, pp. 115–132, 1999.

[33] M. Kim, I. Lee, and O. Sokolsky *Univ. of Pennsylvania, Philadelphia, personal comm.*, 2002.

[34] M. Kim and M. Viswanathan, "Mac: A framework for run-time correctness assurance of real-time systems," *Technical Report MS-CIS-98-37, Dept. of Computer and Information Sciences, Univ. of Pennsylvania*, December 1998.

[35] M. Kim and M. Viswanathan, "Formally specified monitoring of temporal properties," in *Proc. European Conf. Real-Time Systems*, 1999.

[36] I. Lee and H. Ben-Abdallah, "A monitoring and checking framework for run-time correctness assurance," in *Proc. 1998 Korea-U.S. Technical Conf. Strategic Technologies*, 1998.

[37] I. Lee and M. Kim, "Runtime assurance based on formal specifications," in *Proc. 1999 Int. Conf. Parallel and Distributed Processing Techniques and Applications*, 1999.

[38] Y. Liao and D. Cohen, "A specificational approach to high level program monitoring and measuring," *IEEE Trans. Software Eng.*, pp. 969–978, November 1992.

[39] J. Ligatti, "Policy enforcement via program monitoring," *Ph.D. thesis, Princeton University*, 2006.

[40] A. Mok and G. Liu, "Efficient run-time monitoring of timing constraints," in *Proc. Third IEEE Real-Time Technology and Applications Symp.*, pp. 252–262, 1997.
[41] M. Moller, "Runtime assurance based on formal specifications," *Univ. of Oldenburg, Oldenburg, Germany, personal comm.*, 2002.
[42] T. Riegel, "A generalized approach to runtime monitoring for real-time systems," *Master's thesis, TU Dresden*, 2005.
[43] R. D. Russell and M. Chaven, "Fast kernel tracing: A performance evaluation tool for linux," in *Proceedings of the 19th IASTED International Conference on Applied Informatics (AI 2001), Innsbruck, Austria*, February 2011.
[44] C. Watterson and D. Heffernan, "A monitoring approach to facilitate run-time verification of software in deeply embedded systems," *Doctoral thesis, University of Limerick, Ireland*, March 2010.
[45] S. C. V. Raju, R. Rajkumar, and F. Jahanian, "Monitoring timing constraints in distributed real-time systems," in *Proc. Real-Time Systems Symp.*, pp. 57–67, 1992.
[46] M. Saadatmand, M. Sjodin, and N. Ul Mustafa, "Monitoring capabilities of schedulers in model-driven development of real-time systems," *17th IEEE International Conference on Emerging Technologies & Factory Automation (ETFA 2012)*, Sepember 2012.
[47] ITS-EASY post graduate industrial research school for embedded software and systems. http://www.mrtc.mdh.se/projects/itseasy/, Accessed: September 2013.
[48] Combitech. http://www.combitech.se//, Accessed: September 2013.
[49] XDIN AB. http://xdin.com/en/about-xdin/enea-experts/, Accessed: September 2013.

The Impact of Intra-core and Inter-core Task Communication on Architectural Analysis of Multicore Embedded Systems

Juraj Feljan, Jan Carlson
Mälardalen Real-Time Research Centre
Mälardalen University
Västerås, Sweden
Email: juraj.feljan@mdh.se, jan.carlson@mdh.se

***Abstract*—In order to get accurate performance predictions, design-time architectural analysis of multicore embedded systems has to consider communication overhead. When communicating tasks execute on the same core, the communication typically happens through the local cache. On the other hand, when they run on separate cores, the communication has to go through the shared memory. As the shared memory has a significantly larger latency than the local cache, we expect a significant difference between intra-core and inter-core task communication. In this paper, we present a series of experiments we ran to identify the size of this difference, and discuss its impact on architectural analysis of multicore embedded systems. In particular, we show that the impact of the difference is much lower than anticipated.**

***Keywords*—software architecture; model-based analysis; multicore embedded systems; task communication; measurement; cache**

I. Introduction

The majority of computer systems in use today are embedded systems. An embedded system is a microprocessor based system with a typically single dedicated function (as opposed to general purpose computer systems), embedded in and interacting with a larger device. Embedded systems range from simple devices (e.g., MP3 players) to complex systems consisting of multiple nodes communicating over a network (e.g., process controllers), and are used ubiquitously, as we can find them in industry, transportation, medicine, communication, entertainment, commerce, etc.

Today, embedded systems have more complex functionality than ever. At the same time, pieces of functionality that were traditionally realized in hardware are instead implemented in software (e.g., software-defined radio [1]). This makes today's embedded systems increasingly performance intensive. Similarly to general purpose computer systems, there is a trend to tackle the increasing performance demands of embedded systems by increasing the number of processing units, for example by using multicore technology. A multicore processor is a single chip that contains two or more processing units (cores) that are coupled tightly together in order to increase processing power while keeping power consumption reasonable.

Introducing additional processing units increases the performance capacity, but on the other hand introduces the problem of how to best allocate (partition) the software to the available cores, as the allocation has a substantial impact on the performance. A possible way of determining whether a particular allocation of software to cores gives satisfactory performance is to implement, deploy and run the system, in order to collect performance measurements. However, rather than employing such a "fix-it-later" approach, in line with software performance engineering [2], a preferred approach would be to predict the performance with a sufficient accuracy early in the development process, based on architectural models of the system. That way we can get an indication towards good allocations, and avoid time-consuming and costly redeployment of the system when using an iterative measurement-based method. The earlier in the development process that a design fault is caught, the cheaper and simpler it can be fixed. Also, by using models of the system, it is possible to try a large number of candidate allocations in shorter time than by measuring.

In our current work [3], we are investigating an approach for optimizing the allocation of software modules to the cores of a multicore embedded system, with respect to performance. Here, communication time plays a significant role, as it impacts performance aspects relevant in the domain of embedded systems, such as throughput and response time. In a multicore system, the communication time is affected by the allocation of software modules to the available cores. If two communicating software modules run on the same core, the communication normally happens through the local cache and has thus the potential to be much faster than communication between two modules running on different cores, which happens through the shared cache or the main memory. As our work includes design-time model-based performance predictions, we have to take these differences in communication duration into account, in order for the performance predictions to be accurate.

In this paper, we investigate the impact that the allocation of software modules to the cores of a multicore system has on communication time. By performing measurements on a running system, we determine the difference between intra-core communication and inter-core communication under varying conditions. We show that in many situations the difference is significantly lower than we expected, and discuss the reasons and implications of this, namely that the impact of this difference on design-time model-based performance analysis is limited.

The paper is organized as follows. In Section II, we describe the preliminaries and present the motivation for investigating the difference between intra-core and inter-core communication, from the perspective of our current work. In Section III, we give an overview of related work. Section IV is the core of the paper: first it reasons about the expected difference between intra-core and inter-core communication in different scenarios, then it describes the setup of the performed experiment, and finally it gives an interpretation of the results. Section V concludes the paper with a discussion of what the experiment results mean in the context of architectural analysis of multicore embedded systems.

II. Background

The scope of our work are modern (and future) embedded systems whose hardware architecture resembles the one of today's general purpose computers. There is a recent trend of embedded systems moving from single core CPUs to complex multicore CPUs. For example, processors used in today's smartphones and microcontroller boards support up to 4 cores at 2.5 GHz (e.g., ARM Cortex-A15 MPCore [4], Qualcomm Krait 400 [5]).

Typically, each core of a multicore processor has a small on-chip memory (cache), while a larger off-chip main memory (RAM) is shared between the cores. The cache keeps a copy of a subset of data present in the RAM, in order to make this data available to the CPU at a much lower latency than when accessing data from the RAM. For this, the cache utilizes the fact that the same data is often re-accessed frequently (temporal locality of data), and that data being accessed close in time is often stored in adjacent memory locations (spatial locality of data). Other than the local cache (called L1 cache), modern processors typically have additional levels of cache. L2 cache is usually shared between pairs of cores, while L3 cache is shared between all cores. The latency of a particular memory grows in the following order: L1 cache, L2 cache, L3 cache, RAM. Even when having a particular CPU in mind, it is difficult to characterize these values with concrete numbers, but in general L2 cache latency is roughly two to three times larger than L1 cache latency, L3 cache latency is roughly ten times larger than L1 cache latency, and finally RAM latency is two orders of magnitude larger than the latency of L1 cache [6], [7]. When data is transferred between the different memories, it is done in bigger blocks of fixed size called cache lines. A cache line is usually several tens of bytes long.

The software architecture of embedded systems typically consists of a set of concurrent communicating software modules called tasks. The decision of which task to run on which core (i.e., the allocation of tasks) impacts the performance of the system. The extent of the impact depends on the particular performance aspect we consider. For example, schedulability is directly determined by the allocation. If too many tasks are allocated to a single core, the core will be overloaded. As a consequence, tasks will miss their deadlines which is not acceptable for systems with real-time requirements, which embedded systems often have. Similarly to schedulability, it can be expected that task allocation has a large impact on communication time. Two tasks running on the same core can communicate through the L1 cache, while two tasks running on different cores have to communicate through one of the shared memories. This means that intra-core communication should be considerably faster than inter-core communication.

Our current work [3] focuses on optimizing the allocation of tasks to the cores of a multicore embedded system. Already early in the development process, before the implementation, we want to be able to identify the allocations that will result in a system with good performance. We start with an architectural model of the system in terms of tasks and the connections between them, and a model of the hardware platform the system will run on. By an automatic model-to-model transformation, from the architectural and platform models we obtain an executable model of the system, and by simulating this model we get performance predictions for the system. This way we are able to test many allocations in search for the ones that give satisfactory performance. With the term performance, here we mean aspects like throughput and response time. These aspects depend on the communication time, which in turn depends on the allocation of tasks to the cores, as stated above. Therefore, in order to be able to give sufficiently precise performance predictions, we need to identify the difference in communication time depending on whether tasks communicate locally with other tasks running on the same core, or globally with tasks running on different cores. Due to the considerable differences in latencies between the different memories, we intuitively expect this difference to be significant.

III. Related work

Even though the work presented in this paper touches upon research on caches in multicore systems and research on detailed performance evaluations of multicore systems, the context of the work lies in the field of architectural analysis and optimization of embedded systems. We therefore focus the discussion about related work to this research area.

Architectural analysis and optimization of embedded systems can be viewed as a subfield of software performance engineering [2]. Research in this field has a general goal of being able to reason about the performance of embedded systems, already prior to the implementation. At this early stage, embedded systems are typically specified as (more or less formal) models, which can be analyzed or simulated in order to get performance predictions. Often these approaches are complemented with architectural optimization — model-based assessment of particular architecture candidates is enhanced with a mechanism for finding a good architecture. For all but the most trivial embedded systems, evaluating all possible architecture candidates is not feasible, so typically architecture optimization involves a search process aided by heuristics, whose goal is to find near-optimal architectures. In the remainder of this section, we describe several prominent approaches for architectural analysis and/or optimization of embedded systems, both academic and industrial.

ProCom [8] is a component-based and model-based approach for embedded systems in the automotive domain. A ProCom a component is a set of code, documentation, models and extra-functional properties. By utilizing different modeling formalisms, ProCom can analyze worst-case execution times, end-to-end response times and resource usage of embedded systems.

DeepCompas [9] is an analysis framework for predicting performance related properties of real-time embedded systems. The basis of the approach are composable models of individual software components and hardware blocks, which are then synthesized into an executable model of the system. Simulation-based analysis of the executable model results with predicted performance properties for the system. DeepCompas also makes a step towards architecture optimization, by providing support for performing trade-off analysis between several architecture alternatives.

ArcheOpterix [10] is a framework for optimizing embedded system architectures modeled in the Architecture Analysis and Description Language (AADL) [11]. The quality attributes supported by the approach include reliability, performance and energy. One of the key characteristics of the approach is that (through its extension called Robust ArcheOpterix [12]) it takes into account the uncertainty of design-time parameter estimates, and can find architectures that reduce the impact of the uncertainties.

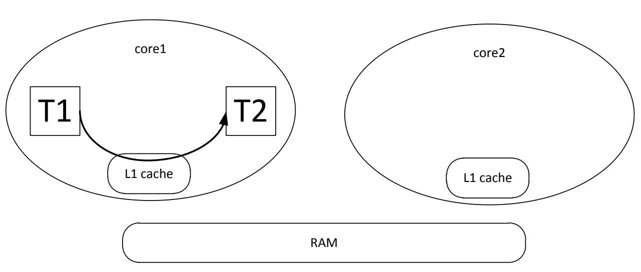
A defacto industry standard for model-based analysis of embedded systems is Mathworks Simulink [13]. It is a graphical tool that comes with built-in libraries of blocks (for instance the Stateflow toolbox for defining and executing state charts) that enable analysis and simulation of embedded systems, and ultimately code generation. It is also possible to define custom blocks using the Matlab programming language, which makes Simulink extendable with custom analysis and simulation techniques.

Additional approaches (not limited to embedded systems) can be found in Koziolek's survey of component-based approaches for performance evaluation [14], and in the survey of architecture optimization approaches by Aleti et al. [15].

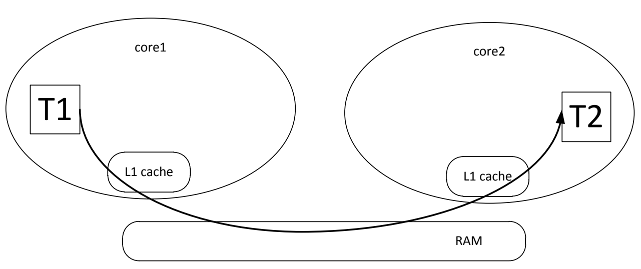
IV. Intra-core vs. inter-core task communication

In this section, we first discuss in more detail about the expected difference between intra-core and inter-core communication in various scenarios. Then, in a separate subsection, we give details about the experiment setup — we describe the hardware and software environments, the general task model and the concrete task setup used in the experiment, and the variation points of the experiment. Finally, again in a separate subsection, we provide an interpretation of the experiment results.

Since many factors other than allocation influence the communication time, such as interruptions from other tasks, we start by identifying the case that has the highest potential of exhibiting a significant difference between intra-core and inter-core communication duration. Imagine the following scenario (Figure 1): a dual-core system, where each core has L1 cache, and the cores share the RAM. There are two communicating tasks: task T_1 produces (writes) data which task T_2 consumes (reads), and task T_2 runs immediately after task T_1 completes. The data fits in the L1 cache. If both tasks run on $core_1$ (scenario depicted in Figure 1a), task T_2 can obtain the data directly from the L1 cache on $core_1$, where it was written when task T_1 produced it. On the other hand, if task T_1 is running on $core_1$ and task T_2 on $core_2$ (scenario depicted in Figure 1b), the data produced by task T_1 is stored in the L1 cache of $core_1$ and not in the L1 cache of $core_2$. So T_2 will



(a) Intra-core communication



(b) Inter-core communication

Fig. 1: Task communication in a dual-core system

have to fetch the data from the RAM. Accessing the RAM is around a hundred times slower than accessing L1 cache, so inter-core communication should be significantly slower than intra-core communication. If the system also had shared L2 cache, the reasoning would still apply — since the latency of L2 cache is around two to three times larger than the latency of L1 cache, the difference in communication times should be smaller than in the case when there is no shared cache, but significant nevertheless.

If two communicating tasks do not run immediately after each other, or if they get preempted by a higher priority task, the data they share might be evicted from the cache, due to other data taking its place. The longer the duration between producing and consuming a particular piece of data, the more likely other data will occupy the cache. In such cases even intra-core communication will have to go through the shared memory, thus reducing the communication time gain from allocating communicating tasks to the same core. Similarly, if the data being communicated does not fit in the local cache, the communication will have to go through the shared memory and the difference between intra-core and inter-core communication is reduced.

A. *Experiment setup*

We use a system with an Intel Core 2 Duo E6700 processor [16]. Each core of this dual-core processor has 32 kB of local L1 cache, while 4 MB of L2 cache is shared between the cores. The cache lines in all caches are 64 bytes long. The system runs the 32-bit version of the Ubuntu 12.04 LTS operating system (kernel version 3.2.29) patched with the PREEMPT RT patch (version 3.2.29-rt44) [17], which turns the stock Linux kernel into a hard real-time kernel. By reducing the overall jitter and enabling the tasks to run at the highest priority, in combination with a high resolution timer of nanosecond granularity, this contributes to reducing unwanted interference in the experiments and increasing the precision of the measurements.

Next, we describe the task model used in the experiments. Tasks are implemented as Posix threads [18], and have read-execute-write semantics, meaning that they first read input data, then preform calculations and finally write output data. A task can either be periodic or event-triggered. A periodic task is activated at regular time intervals, while an event-triggered task is activated when the task it receives data from finishes executing. We assume that the tasks exchange data through shared memory, and that each core has access to the whole main memory. Other models (e.g., distributed memory, where each processor has its own local main memory), are possible but since they are not common in embedded systems, they are out of the scope of this paper.

As identified above, the biggest difference between intra-core and inter-core communication should happen in the case of two communicating tasks that share data which fits into the L1 cache, and the reader task runs immediately after the writer task finishes. We therefore use two tasks in the experiment, a periodic task that writes data, and an event-triggered task that reads the data. The event-triggered task is activated by the periodic task immediately after it has written the data. The data shared between the tasks is an array of integers (integer size is 4 bytes), and each task holds a pointer to it. We use bound multiprocessing, i.e., each task is allocated to a particular core and cannot move to a different core during the execution of a particular experiment. In order to reduce jitter, we run the tasks at the highest priority and prevent memory from being paged to the disk.

In the experiment, we measure the time it takes the reader task to read the shared data. Between the different experiment runs, we vary the allocation of the tasks to the cores, the pattern of accessing the data, and the size of the data the tasks share. Regarding the allocation, in the case of intra-core communication both tasks run on core 1, while in the case of inter-core communication the periodic task runs on core 1 and the event-triggered task runs on core 2.

In order to represent different data access patterns, we vary the stride of accessing the shared data. In other words, the tasks access the data array with different increments (see Figure 2 for an example of different strides; the grey elements are accessed, while the white ones are skipped). In the experiment runs, we use the following strides: 1, 2, 3, 4, 8, 12, 16, 24 and 32. The idea behind using different strides is to compare the reading times in the following cases: (i) when the data is read sequentially (stride 1), (ii) when the data is read nonsequentially with an increment smaller than the cache line (stride 2, 3, 4, 8 and 12), and finally (iii) when the data is read nonsequentially with an increment larger than the cache line (stride 16, 24 and 32).

In a particular experiment run, the writer and the reader tasks access the same amount of data and with the same stride. The amount of data shared between the tasks in different runs is the following number of integers: 128, 256, 512, 4 096, 8 192, 16 384, 262 144, 524 288, 1 048 576, 1 310 720. In order to access N integers with stride S, we allocate a block of data whose size is N * S * 4 bytes. This means that the data we allocate in the different runs varies from 512 B (128 integers with stride 1) to 160 MB (1 310 720 integers with stride 32), and thus covers data that fits into the L1 cache, data that is too large for the L1 cache but fits into the L2 cache, and finally data that is too large for the L2 cache but fits into the RAM.

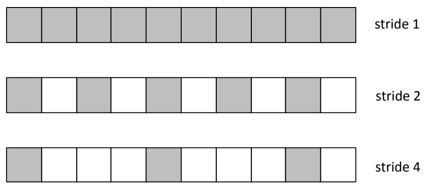
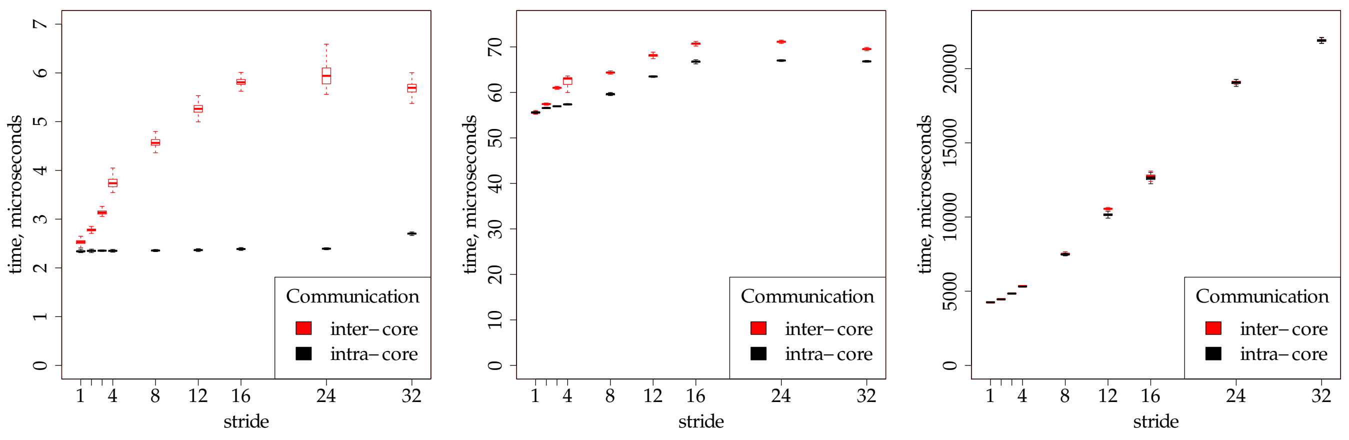


Fig. 2: Stride examples

B. Experiment results

We varied 2 allocations, 9 strides and 10 data sizes, which means that 180 experiment runs were performed in total. In each run, we collected 10 000 measurements of the time it took the event-triggered task to read the data sent by the periodic task. The complete experiment results are available in [19]. Here, we illustrate the results by focusing on three representative data sizes: one that fits into L1 cache (256 elements: from 1 kB for stride 1 to 32 kB for stride 32), one that fits into L2 cache (8 192 elements: from 32 kB for stride 1 to 1 MB for stride 32) and one that fits into RAM (1 048 576 elements: from 4 MB for stride 1 to 128 MB for stride 32). In Figure 3, we show the results as three graphs, one for each data size. As the data size increases, so does the reading time, which is the reason for the difference in the time scales between the graphs. Each graph has two entries for every stride: one for intra-core communication (depicted in black) and one for inter-core communication (depicted in red). Each entry is a boxplot describing the 10 000 measurements. The ends of the boxes show the first and third quartiles, the band inside the box is the second quartile (median), while the whiskers extend to the most extreme data point which is no more than 1.5 times the interquartile range away from the box. For the sake of readability of the graphs, the outliers are omitted.

Comparing the three graphs, we can identify a trend of a relative decrease in the difference between intra-core and inter-core communication when increasing the amount of data shared between the tasks. If we take stride 16 as an example, intra-core communication is 144% faster than inter-core communication when the tasks share 256 integers. When the tasks share 8 192 integers, this difference decreases to 6%, and finally when 1 048 576 elements are shared the difference is 1%. As identified in the beginning of the section, this is expected behavior. If the shared data is bigger than the L1 cache, only the end portion of the data will be present in the L1 cache after the writer task has finished writing the data. Since the reader task reads the data from the beginning, it has to be fetched from one of the shared memories (the L2 cache or the RAM, depending on the size of the shared data), regardless of whether the tasks run on the same core or on different cores.



(a) 256 elements (b) 8 192 elements (c) 1 048 576 elements

Fig. 3: Experiment results

Looking only at the case where the shared data fits into L1 cache (Figure 3a), we see the expected significant difference between intra-core and inter-core communication. The inter-core communication takes roughly three times as long, which corresponds to the difference in latency between L1 and L2 cache. However, the difference is present only at the higher strides. If the shared data is accessed sequentially (stride 1) there is no significant difference between intra-core and inter-core communication. The reason lies in the way data is transferred between cache and RAM — as mentioned in Section II, this is done at cache line granularity. One cache line of 64 bytes corresponds to 16 integers. So even in the case when the data is not present in the L1 cache, as soon as the reader task reads the first integer from one of the shared memories, one whole cache line is transferred to the L1 cache, containing the currently read element and the 15 subsequent elements. Thus, the next 15 elements will be read from the L1 cache. This continues in the same fashion: after reading one element not present in the L1 cache, the next 15 are read from the L1 cache. In other words, in the case of inter-core communication where we intuitively expected 16 cache misses, we got one cache miss followed by 15 cache hits. Increasing the stride increases the share of the elements that create cache misses and decrease the share creating cache hits. This explains the increase of the times it takes to read the shared data in Figure 3a as we increase the stride. When the stride reaches 16, and thus the difference between two read elements reaches the length of the cache line, then reading every element creates a cache miss. The same happens with the strides higher than 16. Therefore, the reading times stay roughly the same even with further increasing the stride. On the other hand, in the case of intra-core communication, the data being read is always present in the L1 cache, regardless of the stride, and the reading times are roughly the same.

V. Conclusion

The experiment confirmed that when tasks share data that is bigger than the local cache, we do not see a significant difference between intra-core and inter-core communication time. On the other hand, when the shared data does fit into the local cache, the experiment only partially confirmed the intuitively expected difference in communication times. Inter-core communication took roughly three times as long as intra-core communication (which conforms with the difference between the latencies of the L1 and L2 caches), but only when the shared data was not read sequentially. In the case of sequential data access, the difference between intra-core and inter-core communication was marginal, due to the way data is transferred between the different memories. It can be expected that data would in fact typically be accessed sequentially, meaning that even in the case of data that fits into the local cache, we would not witness a significant difference between intra-core and inter-core communication.

When the tasks do not share a set of data elements, but rather a very small amount of data (for instance only one integer), then inter-core communication would be significantly slower than intra-core communication. However, this would likely not have a large impact on the response time, since the time it takes to access one data element is typically negligible in comparison with the time that a task spends performing calculations.

In summary, we have seen that the difference between intra-core and inter-core communication in most cases is smaller than what could be anticipated from the difference in the latencies of the local and the shared memory. This was shown for the case when the tasks that share data run immediately after each other, which is the most favorable case for exhibiting a significant difference between intra-core and inter-core communication. A typical application would consist of a set of tasks, meaning that tasks that share data would not always run in immediate sequence, and that the difference between intra-core and inter-core communication would be further reduced.

In the context of design-time architecture-level analysis of multicore embedded systems, this has the following consequences. In order to identify whether a particular case exhibits a significant difference between intra-core and inter-core communication, we need detailed information about data access patterns. This information is typically not available prior to the implementation, when we envision the analysis to be performed. However, as seen from the experiments, in the typical case the difference between intra-core and inter-core communication is not significant enough to hinder performing early performance predictions. Early analysis relies on a set of abstractions and estimates, and for a sufficiently precise performance prediction, a small difference in a particular input to the analysis (in this case the difference between intra-core and inter-core communication time) can normally be ignored.

In the future, we plan to investigate other types of task communication, e.g., message passing. Furthermore, we want to perform a similar experiment on distributed embedded systems, consisting of several interconnected multicore units. Here, the difference between local (intra-node) and global (inter-node) communication should be significant, as intra-node communication uses shared memory, while inter-node communication is preformed over the network.

Acknowledgment

This work was supported by the Swedish Foundation for Strategic Research via the Ralf3 project, and by the Swedish Research Council project Contesse (2010-4276). We would like to thank Tiberiu Seceleanu and Aneta Vulgarakis (ABB Corporate Research, Västerås, Sweden), and Michael Wahler (ABB Corporate Research, Baden-Dättwil, Switzerland) for valuable input and discussions.

References

[1] T. Ulversoy, "Software defined radio: Challenges and opportunities," *Communications Surveys Tutorials, IEEE*, vol. 12, no. 4, pp. 531–550, 2010.

[2] M. Woodside, G. Franks, and D. C. Petriu, "The Future of Software Performance Engineering," in *Workshop on the Future of Software Engineering*, 2007, pp. 171–187.

[3] J. Feljan, J. Carlson, and T. Seceleanu, "Towards a model-based approach for allocating tasks to multicore processors," in *38th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, 2012, pp. 117–124.

[4] Cortex-A15 MPCore, http://www.arm.com/products/processors/cortex-a/cortex-a15.php, [Accessed: 2013-08-20].

[5] Qualcomm Krait 400, http://www.qualcomm.com/snapdragon/processors/800, [Accessed: 2013-08-20].

[6] D. Levinthal, "Performance Analysis Guide for Intel Core i7 Processor and Intel Xeon 5500 processors", http://software.intel.com/sites/products/collateral/hpc/vtune/performance_analysis_guide.pdf, [Accessed: 2013-08-20].

[7] U. Drepper, "What every programmer should know about memory", http://lwn.net/Articles/250967, [Accessed: 2013-08-20].

[8] S. Sentilles, A. Vulgarakis, T. Bureš, J. Carlson, and I. Crnković, "A component model for control-intensive distributed embedded systems," in *Proceedings of the 11th International Symposium on Component-Based Software Engineering (CBSE)*, 2008, pp. 310–317.

[9] E. Bondarev, "Design-time performance analysis of component-based real-time systems," Ph.D. dissertation, Eindhoven Universty of Technology, 2009.

[10] A. Aleti, S. Bjornander, L. Grunske, and I. Meedeniya, "ArcheOpterix: An extendable tool for architecture optimization of AADL models," in *ICSE Workshop on Model-Based Methodologies for Pervasive and Embedded Software*, 2009, pp. 61–71.

[11] SAE standard, no. AS5506, "Architecture Analysis & Design Language (AADL)," 2012.

[12] I. Meedeniya, A. Aleti, I. Avazpour, and A. Amin, "Robust ArcheOpterix: Architecture optimization of embedded systems under uncertainty," in *2012 2nd International Workshop on Software Engineering for Embedded Systems (SEES)*, 2012.

[13] Mathworks Simulink, http://www.mathworks.se/products/simulink/, [Accessed: 2013-08-20].

[14] "Performance evaluation of component-based software systems: A survey," *Performance Evaluation, Special Issue on Software and Performance*, vol. 67, no. 8, pp. 634–658.

[15] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya, "Software architecture optimization methods: A systematic literature review," *IEEE Transactions on Software Engineering*, vol. 39, no. 5, pp. 658–683, 2013.

[16] Intel Core 2 Duo E6700 processor, http://ark.intel.com/products/27251/Intel-Core2-Duo-Processor-E6700-4M-Cache-2_66-GHz-1066-MHz-FSB, [Accessed: 2013-08-20].

[17] PREEMPT RT patch, https://rt.wiki.kernel.org/index.php/Main_Page, [Accessed: 2013-08-20].

[18] POSIX, http://pubs.opengroup.org/onlinepubs/9699919799, [Accessed: 2013-08-20].

[19] J. Feljan and J. Carlson: "The Impact of Intra-core and Inter-core Task Communication on Architectural Analysis of Multicore Embedded Systems — Experiment Results", http://www.idt.mdh.se/~jcn01/research/multicore, [Accessed: 2013-08-20].

Cooperative Optimal Route Planning of Accumulator-Bank Servicing Robots

Ágnes Werner-Stark
and Tibor Dulai
Department of Electrical
Engineering and Information Systems
University of Pannonia
Egyetem str. 10.
Veszprém, Hungary
Email: werner.agnes@virt.uni-pannon.hu
dulai.tibor@virt.uni-pannon.hu

Katalin M. Hangos
Computer and Automation
Research Institute HAS
Budapest, Hungary
and Department of Electrical
Engineering and Information Systems
University of Pannonia
Egyetem str. 10.
Veszprém, Hungary
Email: hangos@scl.sztaki.hu

***Abstract*—Renewable energy storage originating from solar energy is possible in an accumulator-bank, from where the demands and utilization may be provided by robots. A novel optimal route planning algorithm is proposed in this paper that is based on the cooperation of the robots implemented as agents. The interaction between the agent-robots is learned to enhance the stability of the system, and in such a way more efficient operation can be achieved. A special model is developed for describing the operation of the multi-agent system formed by the robots, and the route planning algorithm determines the optimal route considering the cooperation of the robots in special situations. The operation and properties of the proposed algorithm is illustrated using simple examples with robots in different conflict situations.**

Keywords-cooperation; renewable energy; accumulator bank; multi-agent system

I. Introduction

Nowadays, all problems related to the application of renewable energy sources enjoy an increased attention and popularity. Beside of their advantageous properties from environmental and sustainability point of view, these energy sources suffer from limited and unpredictable availability. A solar panel, for example, can produce enough energy during the day, but during the night (or just on cloudy days) it proves unusable. Therefore, a sufficient amount of electrical energy storage capacity should be provided along with each renewable energy source to ensure the availability of sufficient energy on demand. One of the easiest ways is to use accumulators as energy storage, but the price and the storage place they need is too large compared to their capacity. If we would like to store energy in large volumes, we would need to place the accumulators in very large storage parks. The service of this storage (accumulators out and to transport) is a big logistics task. It is then very important to solve the problem of efficient place utilization and the quick service. We developed a system of self-service for an accumulator-bank. For this purpose self-propelled robots are required which are able to transport the accumulators, and can perform independent decision making as well as reacting to certain environmental events. In such a distributed setting, the cooperation among the robots may significantly enhance the performance of the system.

The above accumulator-bank servicing problem is much similar to some well investigated problems in traffic management and control, and logistics. An important approach to solve these problems is to use *multi-agent* techniques. A multi-agent approach to design in the transportation domain is presented in [4]. It presents three important instances for distributed artificial intelligence techniques that proved to be useful in the transportation applications: cooperation among the agents, task decomposition and allocation, and decentralized planning. They can be used to obtain good initial solutions for complex resource allocation problems. As another example, one can consider real-time approaches to manage roadway network congestion over time and space, that is a difficult problem. A solution approach based on cooperative negotiation between agents based on multi-agent principles is proposed in [1].

In one of our earlier works [8], we dealt also with autonomous agents, as we considered such circumstances that make autonomy important, such as extreme high or low temperatures and closeness of dangerous materials. These circumstances had the need of applying robots, they had to solve their problems self-sufficiently, without any direct human intervention.

In the field of logistics, operations research approaches deal with Vehicle Routing Problem (VRP) ([3], [7]) and its solutions that help the companies in their logistic tasks as well. Because of the huge application area of VRP, lots of variants of the problem have born. Some of them include additional constraints (e.g., [9]), while other variants modify the basic tasks (e.g., [5]). The cooperation of vehicles has proven to be useful in this problem class, too [2], where we proposed a method of choosing the directions of the routes of the VRP solution which has the best answer (the minimal extra route) in case of an immediate event supposing cooperative agents. As an immediate event may happen at different phases of the completion of the transportation task, the event's effect has to be taken into account on average.

Usually, in a multi-agent system the agents have specific pre-defined abilities to perform a certain task. One of the challenges of a multi-agent system is to develop agents with the ability of learning from each others' behavior. The aim of this paper is to present an algorithm that allows autonomous agents to use cooperation in conflict situations through communication with other robots. The agents are not in interaction with humans during operation.

The paper is organized as follows. First, we describe the domain of the multi-agent system (Section 2). Then, we introduce our algorithm that can be applied in the context of the multi-robot system example (Section 3). The testing of the algorithm is presented in Section 4. Section 5 concludes the paper.

II. Optimal route planning in the accumulator-bank

This section describes the simple model that is used to design the route of the robots in the accumulator-bank.

A. Plan of routes

Let us divide the storage place into cells of equal size such that a transport robot fits in them. It is very important that we use the available place in the storage the best possible way. The resulted matrix is used as a tool for describing the traffic of the robots: they move from cell to cell to get from one place to another. Obviously, if we use the smallest possible units, then more condition examination and much more calculation have to be performed. There is a trade-off between achieving the best possible result and the efficiency of the algorithm.

Basic assumptions for the route planning algorithm are as follows.

- The capacity of each robot is one unit as is the weight and size of every accumulator, too.
- The orientation is based upon a grid of cells which allows the robots to drive only among the neighboring cells (but not diagonally). In every cell there is at most one robot at a time.
- Every robot moves a unit distance in a unit time, i.e. they move only to the closest neighboring cell. The 90 degrees turn takes a unit time, too.
- One accumulator fits into one storing cell of the storage place.

B. Identification of the optimal route

The robots move along the cells of a grid between the neighboring cells with a one cell per time unit velocity, and the 90 degrees rotation takes a unit time, too. A widely-used path search algorithm has been modified for the identification of the optimal route. This is a popular version of Dijkstra's graph-based algorithm, that was developed by Hart et al. [6], where they described how heuristic information from a problem domain can be incorporated into a formal mathematical theory of graph search and demonstrated an optimal property of a class of search strategies. The algorithm stores the path length from the starting point to the points of graph on the graph's points, that is used again when the recursive algorithm re-visits this point. This re-visit is easily detectable, and the stored value is used to prevent continued counting on the given branch (because we found an existing shorter way) or we can stop the run of the branch because at this point we have already found a more efficient path.

The main modification is that we reduced the cost by reducing the distance between cells. Because the turning of robots requires time, too, we have to record from which direction the robot arrived in to the examined cell and to which direction it continued the search. We add the cost of every 90 degrees turns made between cells as a unit virtual distance. Another modification serves the route which makes traceability easier: when we get a smaller value in a point than the former ones and we overwrite until now the smallest approaching cost, then we note it too, from where (from which direction) the robot comes to a given point. So we can determine easier the compliant route after the filling of a table.

Fig. 1 shows an example of the distance table with the distances in the cells. The green cell is a start point, the blue cell is an end point and the red cells mark obstacles (wall/rack).

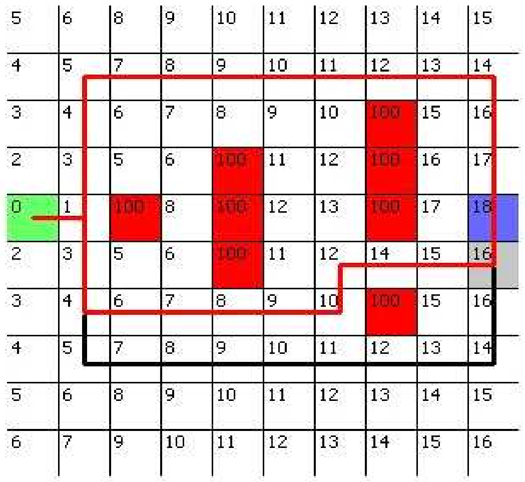


Figure 1. Matrix-based orientation - the problem of listing all the routes

The pseudo code of the proposed basic route search algorithm can be seen in Fig. 2.

The determination of the shortest route is happening backward: it starts at the end point and determines the desired route unanimously. For this purpose one has to record from which cell the robot arrived (source cell) when the length of the shortest route is modified. The easiest method for doing this is to build a new data unit in the cells of matrix (source cell). With this we have a structure similar to a chained list.

It occurs often that there are some routes with equal length between two given points. It is advisable to process each of them, so a crisis situation could be avoided in the future. The routes of equal length present a problem, as the source cell is not enough to store a single value in the cells when the robot can arrive to a cell from several sides after driving the same route length. In order to process the case of equal route lengths properly, it is very important that we consider the following.

- If the robot arrives in a cell and the covered distance is less than the smallest distance until now, we have

```
Date structures:
Map_element: record
                distance: int
                object: wall/empty
                previous: point ((x,y) coordinate pair)
Map: 2 dimensional array, that consists of map_elements

Procedure:
procedure Next(current_point, distance, previous_point)
if current_point is on the map then
    if Map[current_point].object ≠ wall then
        if Map[current_point].distance > distance then
            Map[current_point].distance:=distance
            Map[current_point].previous:=previous_point
            loop for each "new" neighboring cell
                Next(new, distance + 1 + cost of turning, current_point)
            end loop
end procedure
Startup: Next(goal, 0, null)
```

Figure 2. Pseudo code of the proposed basic route search algorithm

to cancel the list of source cells (this information is not relevant).

- If the values of the two distances are equal we have to record it in the source cell to the list, making sure not to overwrite the past values.

C. Constructing the list of all routes

The specification of every optimal route and taking into consideration the turning cost presents a problem when constructing the list of all routes. For clarity, let us consider the example in Fig. 1. The values in the cells mean the distance values and these are determined by the above algorithm. For example, the red lines are the shortest routes (with the same lengths), but the black line has the same length as the other two red lines (not all routes are drawn in the figure). The source of the problem is hidden in the grey colored cell. When the robot approaches the goal from bottom (black line), the distance value of the cell is 17. But when the robot comes from the left-hand side (red line) the distance value is 16, because the cell value is overwritten with the larger 17 value (this information is correct). However, we have to turn 90 degrees to achieve the target following the red line, that would add one time unit at this point and the length of the two routes are in fact equal (17), that we lost by overwriting it.

The easiest way to resolve this problem takes place during the building of the route. We examine all cells and compare the directions either with rotation or a straight route. Since a rotation is not straight, we calculate where the next cell is if we go on straight. If the direction value of this deviates by maximum 1 from the direction value of the current cell then we add this cell to the list of the previous routes. Because in each cell the shortest route to get there and its length are stored, it is enough if we make up the connection only from the overwritten cell, the recursive route construction will bring us to the start cell.

III. Cooperative route searching of the robots

While a single robot navigates in the storage, it can use the previously described algorithm. However, in the case of more than one robot, it is important that we deal with prevention of conflicts, e.g., with the collision of two robots. The possible collision can be detected in advance, not locally. The robots can cross-check the routes in advance so they can search for another route at the start moment instead of waiting for another robot if a possible collision is forecasted. For this reason, the robots make a note to each cell when they pass through it, and notify the other robots about this event. In order to reduce the load of the communication channel, in certain cases the robots may communicate indirectly to each other. In this situation we install a central computer that is able to store the collected information of the storage of the cells and it can pass these information at the request of the robots.

Fig. 3 (a) shows a situation when two robots starting from the points $A1$ and $A3$, respectively, may have a collision.

We have got two possibilities to avoid the collision.

- The robots go to the meeting point and after that one of the robots goes round the other robot. This route will be longer than the pre-planned route because of the turns. This can be seen in Fig. 3 (b).
- If the robots plans their routes in advance then the roundabout route may be shorter, this can be seen in Fig. 3 (c).

When a robot plans a route for itself then it reserves specific cells for itself at pre-planned time instances. Because each navigating robot uses the same time unit, we consider the time unit to be the time step of the system (we suppose that every robot pushes on in synchrony). When the next robot plans its route, it queries the data of the previous robots so it knows exactly what the first robot (or all previous robots) reserved: exactly when and which cells they intend to visit. Now the robot in turn can take this into consideration during route searching, therefore it can decide what is more profitable in case of crossing routes: waiting for the passing of another robot or looking for another route.

A. Waiting for other robots

There may be situations in which it is simply not enough to avoid another robot because for example the robot takes up a bottleneck passage and the other passage is too far. At that time it is more appropriate to wait for the passing of another robot than to choose a bypass route.

In order to handle such situations properly, we should modify the route search algorithm so that the algorithm deals not only with the travelled distance but also with the latency. For this purpose we must note the latency in every cell together with the exact current shortest distance, and when the robot comes into a new cell, we need to compare the sum of the two values with the entered value. If the following cell is reserved at the moment of arrival we must wait until the cell will be empty. During the latency we need to pay attention to the current cell (in which the robot waits) so that no other robot traverses it. If this is to happen, the waiting is not possible.

B. Passages

There can be some narrow passages in the storage for the sake of the better utilization of space, therefore we also need

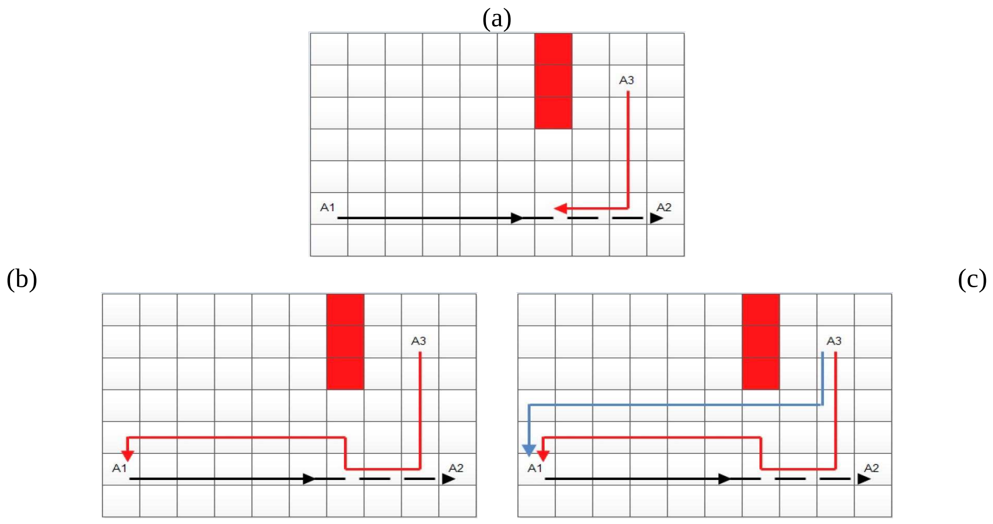


Figure 3. Comparison of the local collision detection and pre-planning: (a) A possible collision, (b) The unplanned route, (c) The pre-planned route

to deal with them. In these passages there can be one robot at a time, this can cause a traffic-jam. If two robots approach the passage at its opposite ends then the route search algorithm can sense only the character of the problem before the collisions.

In order to handle passages in a proper way, a new seizing method had to be developed. When a robot comes into a passage it places a separate seizing at the end of the passage (i.e., at the cell after the last cell of the passage), which is valid not only for the duration when the robot will pass through the cell but it already starts before entering the passage and keeps until when the robot steps out from the passage.

A simple example of a passage situation is seen in Fig. 4. The white squares mark empty cells and the red squares mark obstacles (wall/rack). The robot marked with the blue arrow tries to get out from the passage, his route being reserved. During the route planning of the other robot the recursive algorithm goes in regularly on the red marked route, it senses the collision with the first robot, because this branch stops (in this direction is not any route temporarily). The other green routes are open though, but with different conditions: on the dark green route we have not got to wait for the first robot supposing that we leave the cell before the robot arrives at the end of passage; on the light green route we have to wait in any case for the first robot (or else we find ourselves face to face with the robot and one of them has to turn back). The problem is that the waiting for a given cell (in our case it is $B3$) depends on to which cell we want to go to later.

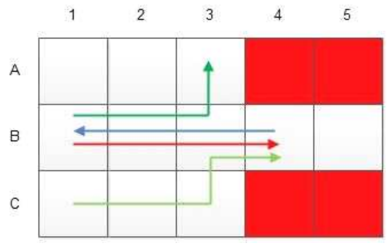


Figure 4. Comparison of the local collision detection and advance planning: The problem of stepping in the passage

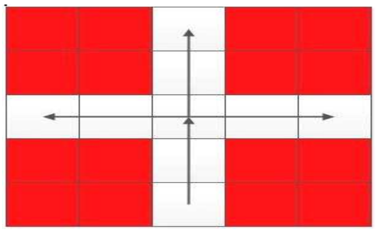


Figure 5. Comparison of the local collision detection and advance planning: Different stopping required in various passages

Unfortunately, however, a more complicated situation can also arise (as it is depicted in Fig. 5. In this case it may happen, that with each of the three different further directions we need to wait for a different duration. This can be resolved if we examine separately every case in the course of route searching. If we perceive a special seizing before entering a cell we have to examine to which passage it is allocated because the rate of waiting will depend on this. For every touched passage we need to create a separate branch and to examine the waiting time of them before we can go over to this special cell. We need to attend to the given branch with the individual waiting time in the direction of only one given cell.

C. Cells multiple visited

There may be cases in which the optimal route passes through a cell twice or even more times. This situation presents a problem to the proposed route planning method, because the cells between two visits can not be clearly defined, and the other robots can not decide on the direction they should proceed. One such example is shown in Fig. 6. The first robot (black arrow) is planning to pass for the first time so the route of this is specific: the robot goes straight from cell E11 to cell E6. If we assume that the robot can pass through a cell only once, the second robot (red arrow) is forced to make a long roundabout way. The shortest route will be the blue route, i.e. the robot shuns the front of the other robot in cell F7 and waits while the other robot passes and then continues on it is way. The original route planning algorithm can not process

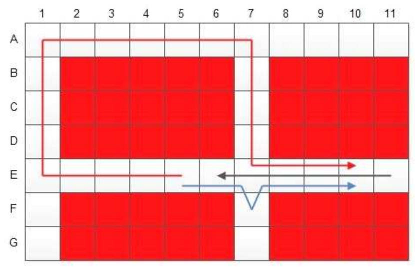


Figure 6. Comparison of the local collision detection and advance planning: The strategy of the stand aside option in contradiction to the roundabout way

this route satisfactorily, because we leave the end points of the passages (here it is cell E7) and we register the following information: the previous value of the cell F7 will be E6 and the previous value of the cell E8 will be F7. Thus, there will be a stoppage in the course of decryption of the route.

A similar problem may arise where a robot decides to pass through the cell E7 twice because there are two different distance values and waiting values. The different waiting value case is more important. When the second robot passes through the cell for the first time, the algorithm records 0 waiting, then a positive value for the second time (assuming that we need to wait some time units for the first robot). These values should also be noted separately in a suitable data structure. Three parameters are needed for this: the previous and next cells (these identify where the robot came from and where it is going to when it passes through the cell) and the waiting value. This permits us to record the difference in the duration of waiting times before the robot steps into the different passages.

IV. Case studies

Simple case studies were used to test the operation and efficiency of the proposed cooperative route planning algorithm. For this purpose an implementation of the algorithm has been developed in Delphi programming language (we used also Indy (Internet Direct) component package to the communication), and this simulation environment can visualize the robots' movements.

Every measurement result was verified with the following configuration:

- CPU: Inter Celeron 560@2.13 GHz
- Operating system: Microsoft Windows XP SP2

A. Route planning tests

Each case study had a few cooperating robots and the topology of the accumulator-bank storage place was also different. The planning order of the robots was the same in each case, and it corresponded to their serial number (in ascending order).

Fig. 7 illustrates the starting situation and the movements of the robots in the following two examples. The grey cells show the actual positions of the robots, where both the robots' serial number and their actual direction are indicated. The cells with a number denote the goal of the robot with the same serial number.

TABLE I. The effect of the map size on the running time

Size of map	Time of route planning (ms)	Time of planning/ robot(ms)
25x25	62,6	3,13
25x50	175	8,75
50x50	334,8	16,74

TABLE II. The effect of the robot number on the running time

Number of robots	Time of route planning (ms)	Time of planning/ robot(ms)
5	14,4	3,08
10	28	2,8
15	46,8	3,12
20	62,6	3,13

- **Example 1: Passages**
  There are four robots in the storage and they know to which cells they need to get to. We can see in Fig. 7 (a) that every robot stands in compliance with his forward direction. Robot3 waits till robot1 and robot2 pass through the passage, thereafter robot3 goes on in the direction of its goal. Robot4 has attained the goal in the meantime because its route has not crossed the others. Fig. 7 (b) illustrates the movement of the robots in this situation.
- **Example 2: Getting out of the way**
  This example illustrates the getting out of the way: robot1 planned first, it has priority, so robot2 gets out of its way in the other passage. Thereafter robot2 continues on its way when robot1 passes before it. We can see the starting situation in Fig. 7 (c), and the movements in Fig. 7 (d).

B. Efficiency test

In order to test the efficiency of the proposed algorithm, we recorded the full running time of the algorithm and noticed how this value changed with the increasing complexity of the planning problem.

Effect of the map size: In the first test we examined how the time of planning changes by increasing the size of the map applying the same number of robots (in the present instance 20 robots). The robots were randomly placed on the map. The results are collected in Table I.

The average length of the randomly placed robots' route doubled in case of doubling the map size, so the route search algorithm had to explore the space with twice as large radius in this case.

Effect of the number of robots: In case of the other test the robots were arranged randomly in a $25x25$ of size map-file. Five program running was performed with each robot number value, and the running times were averaged. Table II shows the simulation results.

It can be seen from the results that the system integrates the new robots well, the robot pre-planning time is about 3 ms independently of the number of robots. This important result

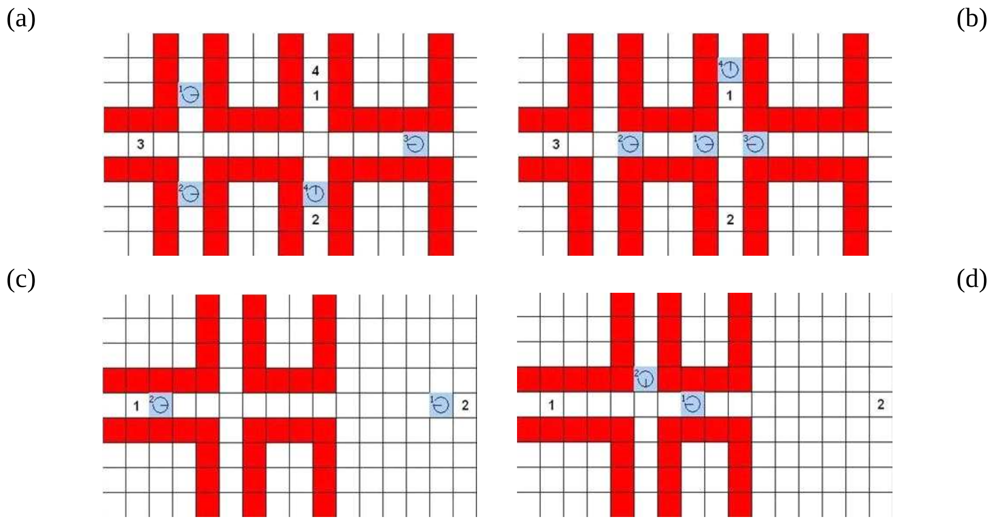


Figure 7. (a) Example1, starting situation, (b) Example1, movements of the robots, (c) Example2, starting situation, (d) Example2, movements of the robots,

shows that the proposed algorithm scales up well with the size and complexity of the problem, thus offering an efficient service of the accumulator-bank.

V. Conclusion

A novel optimal route planning algorithm is proposed in this paper that is based on the cooperation of the robots implemented as agents. The basic version of the algorithm uses a special data structure that is arranged according to the matrix-type grid of the cells defined in the storage place.

The robots use the same route planning algorithm in turn, and take into account the plans of the other robots in order to avoid collision. This way they can detect and avoid collision in advance and not locally. Special conflicting situations, including waiting, passage handling and multiple visiting of cells are also investigated.

The operation and properties of the proposed algorithm are illustrated using simple examples with robots in different special conflict situations.

For optimizing the navigation of the robots, we aim at enriching their communication process with learning capabilities.

Acknowledgment

We acknowledge the financial support of the Hungarian State and the European Union under the TAMOP-4.2.2.A-11/1/KONV-2012-0072. This publication/research has been supported by the European Union and Hungary and co-financed by the European Social Fund through the project TAMOP-4.2.2.C-11/1/KONV-2012-0004 - National Research Center for Development and Market Introduction of Advanced Information and Communication Technologies.

References

[1] J.L. Adler and V.J. Blue, ”A cooperative multi-agent transportation management and route guidance system”, Transportation Research Part C: Emerging Technologies 10(5-6). 2002, pp. 433-454.

[2] T. Dulai and Á. Werner-Stark, ” Immediate event-aware routing based on cooperative agents”, Proceedings of Factory Automation 2012, Veszprém, 21-22 May. 2012, pp. 144-148.

[3] B. Eksioglu, A.V. Vural, and A. Reisman, ”The vehicle routing problem: A taxonomic review”, Computers & Industrial Engineering 57, 2009, pp. 1472-1483.

[4] K. Fischer, J.P. Müller, and M. Pischel, ”Cooperative transportation scheduling: An application domain for DAI”, Applied Artificial Intelligence: An International Journal, 10(1), 1996, pp. 1-34.

[5] M. Gendreau, F. Guertin, J.Y. Potvin, and R. Séguin, ”Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries”, Transportation Research Part C 14, 2006, pp. 157-174.

[6] P.E. Hart, N.J. Nilsson, and B. Raphael, ”A formal basis for the heuristic determination of minimum cost paths”, IEEE Transactions on Systems Science and Cybernatics, 1968, pp. 100-107.

[7] G. Laporte, ”The vehicle routing problem: An overview of exact and approximate algorithms”, European Journal of Operational Research, No. 59, 1992, pp. 345-358.

[8] Z. Szabó, B. Lájer, and Á. Werner-Stark, ”Automata Depository Model with Autonomous Robots”, Acta Cybernetica 19, 2010, pp. 655-660.

[9] D. Zhenggang, C. Linning, and Z. Li, ”Improved Multi-Agent System for the Vehicle Routing Problem with Time Windows”, Tsinghua Science and Technology, 14(3), 2009, pp. 407-412.

Business Architecture for a SME: A Case Study of a Manufacturing Firm in Mexico

Alicia Valdez
Research Center
University Autonomous of Coahuila
Coahuila, Mexico
aliciavaldez@uadec.edu.mx

Carlos Vega, Elias Olivares, Juan Perez
Research Center
UPAEP
Puebla, Mexico
carlosarturo.vega@upaep.mx,elias.olivares@upaep.mx,
juancarlos.perez@upaep.mx

Abstract— **Enterprise architecture is a subject that has increased its importance in the Small and Medium Enterprises in the manufacturing sector of the industry in Mexico. The global competitiveness of the markets has influenced the adoption of methodologies that support the strategic alignment of the processes with the goals and strategic objectives of the firms. The components of the business architecture like mission, vision, strategic objectives, products, organizational structure, business processes, clients and geographic region, were collected from the firm of the case study for the design of the architecture. As a result of the practical application, an implementation model has been created and four strategic objectives were established for to improve productivity and competitiveness. This paper is a result of the research project of analysis, design and implementing business architecture in a medium size manufacturing company like partial architecture of an enterprise using ontologies for representing the core elements of the business architecture; the study presents clearly the importance of the strategic planning for the analysis and the detection of the main faults for the success of the achievement of goals and objectives.**

Keywords-Business architecture; SME; Enterprise architecture; Key processes

I. INTRODUCTION

Five major markets are emerging in the world, namely, China, India, Southeast Asia, Latin America and Eastern Europe, where global manufacturing companies have considered to make investments, because these regions are rapidly growing economies with great potential for business [1].

In these regions, the Small and Medium Enterprises (SMEs) are important in the development of the economy as they have a great capacity of generating jobs. In Mexico, 99% of the companies are SMEs, (for every 10 employees, 7 of these are working on SMEs) [2]; the study “Impact Evaluation of SME Programs in Latin America and the Caribbean”, developed by the World Bank has mentioned some of the problems faced by SMEs, among which are [3].

- Access to financing;
- Weak management capacity;
- Lack of ability to exploit economies of scale in production;
- Poor information about market opportunities, and
- New technologies and methods of work organization.

The Enterprise Architecture (EA) is a strategic solution to improve the capabilities of these companies and respond quickly to the challenges, either business related or technological which is today’s markets demand.

EA is also a way that aims to provide companies with a framework for the use of information on business processes in ways that support the business strategy [4]. Orantes, Gutierrez, and Lopez have mentioned that the company should be in a constantly evolving, redefining business processes, to achieve business process architecture, which is the basis for subsequent architectures [5].

The EA is the instrument that establishes the structure of the company, is a conceptual model of the business and information technology solutions (IT), seen as a set of pieces that involves processes, and functions that works together in a coherent and well defined way [6].

Some authors consider that SMEs have lesser tendencies to use IT for strategic purposes [7], and the success of architectures implementation depends on consistent objectives between IT strategy and business strategy [8].

The Business Architecture (BA) is a partial architecture of the EA, where the business is defined, the organizational structure is documented, and the business processes are identified.

BA analyses the business model relying on strategic planning with their areas of interest [9].

In this case study, an analysis was performed to establish: What key processes in manufacturing SMEs are included in the EA design, as well as, the practices and business modeling tools that use these companies to develop EA; with the objective of supporting them in increased productivity and competitiveness. A proposal was developed that included EA standards, software tools, and methods [10].

The software tools [11] can provide support for this particular type of companies, that can be easy to use and implement to help them in their process of establishing enterprise architectures, while developing, a parallel process of Strategic Planning to support in setting goals and strategic objectives.

The organization of this paper is as follows; first section is mentioning the concepts and methods of the EA; the second section shows the information required for the business architecture from the company; finally, the implementation of the solution derived from the analysis is shown.

II. CONTENT

The EA started as management information systems in the early 60's in the United States Company International Business Machines (IBM) by “Information Systems, Control and Planning Staff" (ISCM) area.

The methodology known as Business Systems Planning (BSP) was considered one of the methodologies that started the EA.

John Zachman, who worked at ISCM, developed a framework for defining the architectures of information systems, subsequently became the "Zachman Framework" [12]; one of the perspectives was the business model of the company.

In 1994, the Department of Defense of the United States of America [13] created the Technical Architectural Framework for Information Management (TA-FIM), based on Zachman framework.

In 1996, the Congress of the United States of America passed a law called "Clinger-Cohen Act of 1996", which specifies that federal agencies should improve the efficiency of investment on information technologies, establishing the Council of Managers of information Technology (CIO's Council) group, which originated the Federal Enterprise Architecture Framework (FEAF) [14].

TAFIM was withdrawn by the Department of Defense and the association donated to The Open Group, who later developed The Open Group Architecture Framework (TOGAF) standard [15].

TOGAF is an enterprise architecture methodology and framework, used in organizations to improve business efficiency [16], based on the Architecture Development Method (ADM); ADM is divided into 9 phases, an overview of the architecture describing how the new capacity going to align business goals and strategic objectives with IT. Fig. 1 shows the phases of the ADM Method.

In the firsts two phases, preliminary and “A”, the principles of the architecture and the architecture vision are defined.

In the “B” phase, the BA with the fundamental business organization and its goals, objectives, business processes, functions, services, human resources, organizational structure, the principles governing its design and evolution are analyzed.

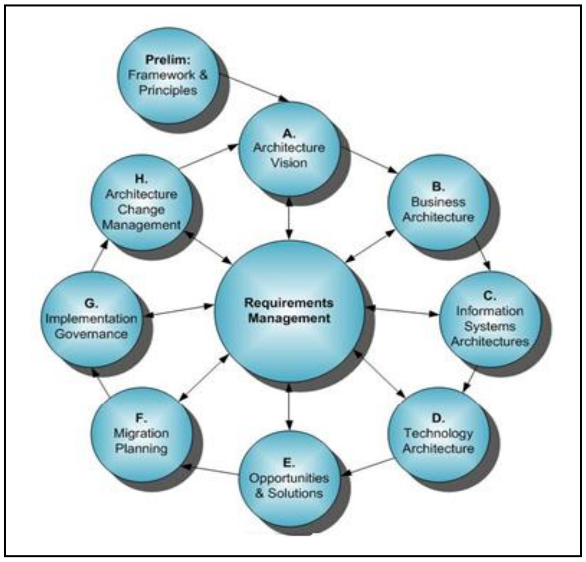

Prelim: Framework & Principles
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities & Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

Figure 1. Architecture Development Method Phases

The metal mechanic industry is representative of the northern of Mexico, which provides raw material to the automotive cluster with some important firms like General Motors, Chrysler, and other important companies.

The suppliers of the cluster are mainly SMEs; this case was developed in SME of the metal mechanic industry and the flow of the information of the study case is shown in Fig. 2.

Beginning with the collection of the information required, this information was captured in the ontology editor; from the editor were obtained reports and maps, and flowcharts of the processes were built; after these activities, the design of the BA was realized, and finally, the implementation of the design.

The results showed some opportunity areas for improvement, in the company.

A. Information of the Business Architecture

The BA involves some elements of the company like mission, vision, objectives, goals, values and policies, business processes, procedures and functions, organizational structure, situational analysis, customers, markets, products and long, medium and short strategies.

Tables I and II show the data of the BA elements, processes like distribution, finance, human resources, production, quality, sales and marketing, information technology, and product development.

Each process has a set of activities; for example, the product development includes production cycle program, cutting, marking, machining and forming of steel plates, and profiles. All processes were collected from the company and recorded in software tools.

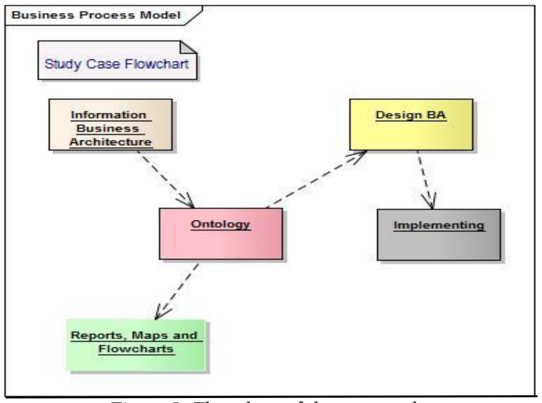


Figure 2. Flowchart of the case study

This information served as base for the next architectures, application and technology, where each process are linked with a software application and technology that supported it.

TABLE I. DATA FOR THE BUSINESS ARCHITECTURE

Business Architecture	Description	
Mission	"Serve society, customers, employees, suppliers, be the best option for all".	
Vision	"Being a quality supplier of metal products, broaden participation in national and international markets"	
Values	Responsibility, loyalty, respect and quality production.	
Objectives	Improve the relationship between customers and suppliers. Minimize operation failure. Maximize the performance of the raw material.	Have better management control. Improve planning processes. Investment plan in machinery and equipment.

The organizational structure of the company has 4 levels corresponding to the position of the Chief Executive Officer (CEO) and sales manager for the level 1; head of production machining, head buyer, finance officer, and human resources manager for level 2; machining supervisor, pailer supervisor, warehouse manager, billing, and quality control for level 3; machining operators and pailer operators for level 4.

The customers belong to the local market of the northern of Mexico with Altos Hornos de Mexico, S.A. (AHMSA), TAKATA Industries, General Motors Company, Chrysler, and other companies of the metal mechanic industry.

The market is regional; the firm can compete in global markets adopting a strategy of certification in quality processes. The products of manufacturing are: General forklift parts, rotatory joints, various pieces of mechanical equipments, and assembly using Computer Numerical Control (CNC) machines.

The first strategy is to manufacture products with high quality that markets demands, the support of the IT can permit to reduce costs, and increase the competitiveness and the productivity.

With the data obtained from the company, the next step is entering data in the ontology software; for this case, we use the free software Protégé Ontology Editor 3.4 [17], developed by the Stanford University; in this software, we can have one super class called EA with some subclasses like business architecture, information architecture, applications architecture, and technology architecture.

The main components of the BA class are shown in Fig. 3, these are:

TABLE II. PROCESSES OF THE COMPANY

Company area	Processes	
Distribution	Finished products delivery	
Finance	Management company's finances	
Human Resources	Personnel administration	Detect training needs of business areas, especially productive areas for develop entrepreneurial training program
Investment Administration	Investments of the company	
IT	Provision of IT support for company's business processes	
Quality	Manufacture that meets production specifications	Testing and inspection using ultrasonic methods or industrial inspection
Sales and Marketing	Management customers. Customer service	Continuous communication with customers to identify needs and complaints.
Stock	Register the inputs and outputs of goods and raw materials.	Suppliers management.
Product development	Program production cycles	Cutting, marking, machining and forming of steel plates and profiles

- Objective (strategic alignment);
- Principle (production requires all processes);
- Domain (distribution, sales, quality, etc);
- Role (CEO, chief of sales, pailer operator);
- Capability (planning sales, machinery operator);
- Product (rotatory joints, machining), and
- Process (design products, sales management).

The BA objective is the strategic alignment between business goals and IT represented like “Strategic Alignment.”

The business domains are all areas and functions of the company, like distribution, finance, sales, and others.

The principle business is “Production requires all processes.” Business roles are performed by people and the business process are all the processes represented in Table II, required for the company operation.

The graphical representation of the all BA components is the link between the collected data and the software tool.

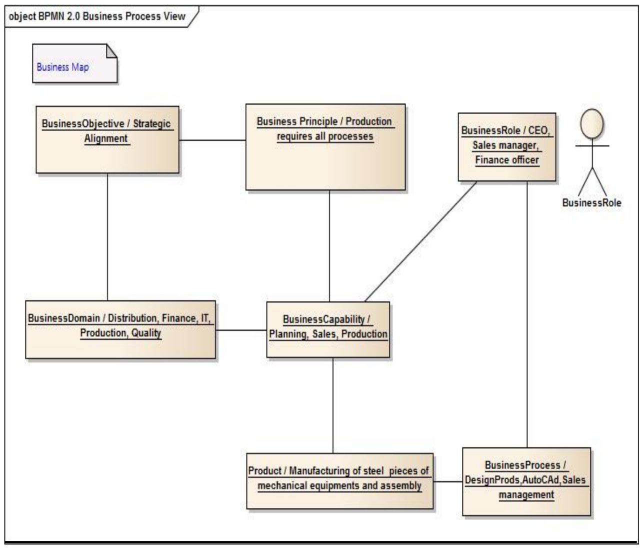

object BPMN 2.0 Business Process View
Business Map
BusinessObjective / Strategic Alignment
Business Principle / Production requires all processes
BusinessRole / CEO, Sales manager, Finance officer
BusinessRole
BusinessDomain / Distribution, Finance, IT, Production, Quality
BusinessCapability / Planning, Sales, Production
Product / Manufacturing of steel pieces of mechanical equipments and assembly
BusinessProcess / DesignProds,AutoCAd,Sales management

Figure 3. BA Design

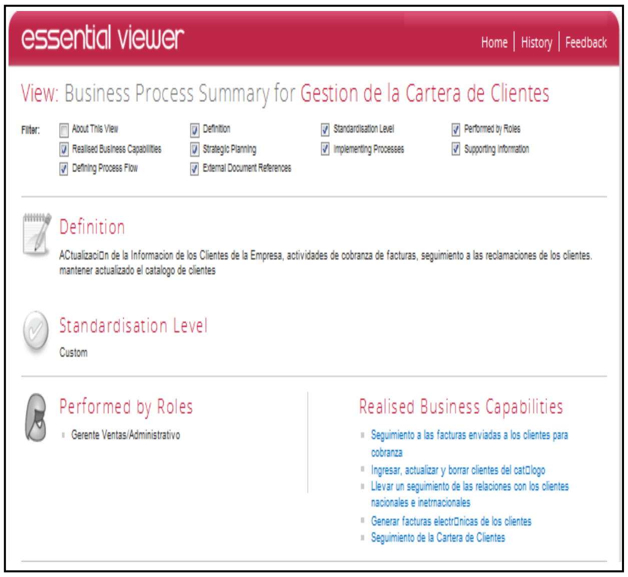
Fig. 4 displays the customer’s process with roles and capabilities, after entering data in the ontology editor.

The information of the editor is sent to a graphical tool [11], functioning like repository, which is an open source software tool for the management of AE, this tool is Essential Architecture Manager 3.0 [18], requires some prerequisites software like Apache Tomcat 5.5 or above, Java Runtime 1.5 or above, Graphviz 2.26, and the Protégé Ontology Editor 3.4 or above.

This set of tools creates a graphical environment for representing the EA, and each of the partial architectures, like BA. Fig. 4 shows partially one process with the components of the BA for this process.

All the processes were represented in the software tools for purpose of completing the BA for subsequent architectures.

B. Implementation model



View: Business Process Summary for Gestion de la Cartera de Clientes

Filter: [ ] About This View [✓] Definition [✓] Standardisation Level [✓] Performed by Roles [✓] Realised Business Capabilities [✓] Strategic Planning [✓] Implementing Processes [✓] Supporting Information [✓] Defining Process Flow [✓] External Document References

Definition

ACtualizaci□n de la Informacion de los Clientes de la Empresa, actividades de cobranza de facturas, seguimiento a las reclamaciones de los clientes. mantener actualizado el catalogo de clientes

Standardisation Level

Custom

Performed by Roles

- Gerente Ventas/Administrativo

Realised Business Capabilities

- Seguimiento a las facturas enviadas a los clientes para cobranza
- Ingresar, actualizar y borrar clientes del cat□logo
- Llevar un seguimiento de las relaciones con los clientes nacionales e inetrnacionales
- Generar facturas electr□nicas de los clientes
- Seguimiento de la Cartera de Clientes

Figure 4. Example of process in software tool

The implementation model is centered on the client, the material resources and equipment, organization, human resources, logistics, and all that the company needs for working in his goals and objectives; so the circle represents the firm, the architectures are around the circle, and are applied to all processes, for to identify strategic changes with assessment of options tending to produce a change plan. Fig. 5 shows the implementation model described.

Some components of the BA were redesigned as a result of the analysis for implementation updating:

- Mission;
- Vision;
- Strategic objectives, and
- Organizational structure.

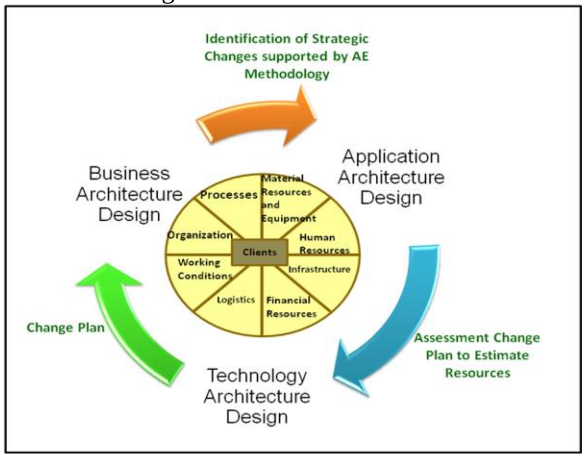

Identification of Strategic Changes supported by AE Methodology
Application Architecture Design
Business Architecture Design
Processes
Material Resources and Equipment
Organization
Clients
Human Resources
Working Conditions
Infrastructure
Logistics
Financial Resources
Change Plan
Assessment Change Plan to Estimate Resources
Technology Architecture Design

Figure 5. Implementation model diagram

Considering full knowledge of the manufactured products, identifying potential markets, and their competitive advantage, and the vision determines the strategic direction of the company. Four strategic objectives were established:

Strategic objective 1: Increase production and competitiveness to achieve better sales and increase company revenue.

Strategic objective 2: Update, acquire, and implement the technology required for increased production and competitiveness.

Strategic objective 3: Update recruitment processes and training of existing staff to increase integration and productivity.

Strategic objective 4: Secure your position with existing customers, increase local sales and find new customers in global markets.

Two areas from the organizational structure were added: Human Resources and Logistics; the justification was that the firm does not have human resources area for the training of the employees, and logistics are required for the management of the resources from the beginning of the value chain to final assembly.

Other needs identified like the strengthening of the market position, the total quality culture, and the training of human resources to achieve improved organizational climate and consequently on the productivity of the entire company.

Current management skills are not sufficient for the next five or ten years; it requires that managers, although have professionals studies in engineering, must be kept updated on the latest management techniques.

Competition modernizes its production techniques and new companies emerge, so the upgrading of equipment and technological infrastructure is vital to the long-term performance of the company.

III. CONCLUSION

The process of developing a BA in a SME in metal mechanic industry in Mexico reflects the needs of this industry sector, to upgrade their management skills to compete in global markets.

BA must be focused on the importance of the technology strategy aligned with the business strategy, to gain competitive advantage from the use of IT, to enable them to be inserted into global markets with a clear plan.

This project helped to meet the needs of SMEs companies to propose affordable solutions that make business management resources and technology to solve problems.

BA can be represented on a business ontology designed especially to support the structuring of architectural maps of the company and its relationships with strategic objectives.

It is necessary to comprehensively conceptualize strategic planning of the company to continue with the design of the ontology in the subsequent phases as application and technology.

The contribution of the paper focuses on the approach to the problems of the company, with the help of business architecture, software tools, and the implementation model.

The BA supports the strategic alignment between objectives, goals, and business processes.

In the future, this industry will be supplier of the aerospace industry in the country, and would be integrated to a more specialized chain with greater scope in the domestic and international markets.

This research project was developed in one year for a doctoral dissertation in Strategic Planning.

REFERENCES

[1] Deloitte T., ”Innovation in emerging markets 2007 annual study”, http://www.deloitte.com/assets [retrieved: 06-2013].

[2] Secretary of Economy, ”SMEs news”,http://economia.gob.mx/ [retrieved: 04-2013].

[3] G. Lopez and H. Tan, “Impact evaluation of SME programs in Latin America and the Caribbean”, World Bank, Washington, USA, 2010, pp. 4-10.

[4] S. Spewak and S. Hill, ”Enterprise architecture planning, developing a blueprint for data, application and technology”, Wiley publisher, USA, 1992, pp. 1-6.

[5] S. Orantes, A. Gutierrez and M. Lopez,”Enterprise architectures: Business processes management vs Services oriented architectures, are they related?”, Redalyc, vol. 13, pp. 136-144, 2009.

[6] W. Bruls, M. Steenbergen, R. Foorthius, R. Bos, and S. Brinkkemper, “Domain architectures as an instrument to refine enterprise architecture”, Communication of the association for information systems, vol. 27, pp. 517-540, 2010.

[7] B. Goh, “Applying the strategic alignment model to business and ICT strategies of Singapore’s small and medium sized architecture, engineering and construction enterprises”, Construction management and economics, vol. 25, pp. 157-169, 02-2007.

[8] H. Voordijk, A. Leuven and, A. Laan, ”Enterprise resource planning in a large construction firm: implementation analysis”, Construction management and economics, vol. 21, pp. 511-521, 2003.

[9] S. Spewak, “Enterprise architecture planning”, Wiley publisher, 2000, pp. 85-88.

[10] J. Schekkerman, “Enterprise architecture good practices guide: How to manage the enterprise architecture practice”, Traffod publisher, 2008, pp. 15-20.

[11] D. Rice, “Review of essential architecture manager 1.0”, Journal of enterprise architecture, vol. 1, pp. 1-7, 05-2009.

[12] J. Zachman, “A framework for information systems architecture”, IBM sytems journal, vol 26, pp. 276-292, 1987.

[13] Departament of Defense of the United States of America, ”DoD architecture framework version 2.0”, 2009, http://dodcio.defense.gov/dodaf20.aspx [retrieved: 05-2013].

[14] R. Sessions, “A comparison of the top four enterprise architecture methodologies”, MSDN Library, http://msdn.microsoft.com/en-us/library/bb466232.aspx [retrieved: 03-2013].

[15] The Open Group, ”TOGAF”,http://www.opengroup.org/togaf/ [retrieved: 05-2013].

[16] The Open Group,"Management overview, in ADM basic principles,http://www.togaf.info/ [retrieved: 05-2013].

[17] Stanford University, "Tutorial documentation of protégé", http://protege.stanford.edu/doc/users.html#tutorials [retrieved: 05-2013].

[18] Essential project,"The Essential Project", 2013, http://www.enterprise-architecture.org/ [retrieved: 06-2013].

Confirming Design Guidelines for Evolvable Business Processes Based on the Concept of Entropy

Peter De Bruyn, Dieter Van Nuffel, Philip Huysmans and Herwig Mannaert
Normalized Systems Institute (NSI)
Department of Management Information Systems
University of Antwerp
Antwerp, Belgium
{peter.debruyn, dieter.vannuffel, philip.huysmans, herwig.mannaert}@uantwerpen.be

***Abstract*—Contemporary organizations need to be agile at both their IT systems and organizational structures (such as business processes). Normalized Systems theory has recently proposed an approach to build evolvable IT systems, based on the systems theoretic concept of stability. However, its applicability to the organizational level, including business processes, has proven to be relevant in the past and resulted a.o. in a set of 25 guidelines for designing business processes. In subsequent work, the Normalized Systems theory was confirmed and extended based on the concept of entropy from thermodynamics. Therefore, this paper explores whether the guidelines which have been proposed for business processes from an evolvability point of view can be confirmed or extended from the entropy reasoning as well. More specifically, the validity of 9 business process design guidelines is investigated for this purpose. Our results indicate that the investigated guidelines are rather consistent among both approaches: guidelines required to attain evolvability seem to enable low entropy (i.e., complexity) and vice versa.**

***Keywords*—Business Processes; Complexity; Entropy; Normalized Systems.**

I. INTRODUCTION

Lack of organizational agility is often attributed to a lack of IT agility [1] as IT systems ensure the support or even automation of business processes. Consequently, organizational changes need to be reflected in both the business processes and their supporting information systems. This means that, instead of focusing solely on IT systems, attention for the design and agility of the business processes is needed as well. The explicit attention for the design of business processes emerged when the implicit work practices were automated using ERP systems [2]. It was recognized that the hard coding of the business processes in software packages resulted in a lack of adaptability of the processes [3]. As a result, the design of business processes gained a central role in organizations, separated from the design of information systems [2]. However, integration of business processes and information systems still needs to be achieved, and agility (or "evolvability") needs to be ensured on both levels.

Normalized Systems (NS) theory offers a theoretically founded way to design software systems which exhibit evolvability based on the systems theory's concept of stability, by proposing a limited set of design theorems [4], [5]. Applying the theory's rationale to the business process level has been shown feasible and resulted a.o. in a set of 25 guidelines for designing evolvable business processes [6]. In subsequent work, NS theory was confirmed and extended based on the concept of entropy from thermodynamics [7]. This extension resulted in additional theorems, while confirming the existing theorems. Therefore, it might be interesting to verify whether the guidelines which have been proposed for business processes can be confirmed or extended from the entropy reasoning as well. This paper explores this research area by applying the entropy reasoning to a set of business process guidelines (which were originally proposed to design evolvable business processes). First, we provide some theoretical background (Section II). Afterwards, the guidelines (Section III) and discussion (Section IV) are presented. Finally, our conclusions are offered in Section V).

II. THEORETICAL BACKGROUND

NS was introduced as a theoretically founded way for deterministically designing software architectures exhibiting a proven amount of evolvability. For this end, the systems theoretic concept of stability is applied [4], [5]. This implies that a bounded input function (e.g., "add data attribute") should result in bounded output values, even as time $T \rightarrow \infty$. It has been proven that at least four theorems should be consistently applied in order to obtain such evolvable software architecture [4], [5]. Violations against these theorems can be observed at compile-time [5].

Later on, the theory has been proven to be applicable to the design of evolvable business processes [6]. Here, business processes are considered at their most elementary level (i.e., the "elementary tasks and elementary sequencing and design of these tasks"). To obtain stability, it is required that changes to individual processes or tasks do not impact other processes or tasks [6]. In order to achieve such Normalized Business Processes (NSBPs), a set of 25 guidelines was developed, based on the four NS theorems [6].

In order to position this research, a clear distinction between the concepts evolvability and flexibility is necessary. Although flexibility also denotes a desired characteristic of business processes, as defined by e.g., [8]: '*the capability to implement changes in the business process type and instances by changing only those parts that need to be changed and keeping other parts stable*"; it differs from evolvability defined as the capability of a modular business process design to adapt to identified change drivers [6]. It also differs from the change patterns research, as that research focuses on how (operationally) processes should be changed to be flexible, whereas this research focuses on why and how processes should be (structurally) designed in order to support change. Matching the flexibility types of Schonenberg [9], evolvability can be situated within the *Flexibility by Design* type. Nevertheless, designing evolvable business processes actually precedes flexibility as run-time (flexible) design decisions should comply with the requirements of evolvable business processes at design time.

In subsequent research, NS was extended based on the thermodynamic concept of entropy, initially focusing on software architectures again [7]. As entropy is generally associated with concepts as complexity, amount of disorder or available information, it enables the study of the diagnostability of a (software) system. In statistical thermodynamics, entropy is considered proportional to the number of microstates consistent with one macrostate (i.e., its multiplicity) [10]. The macrostate refers to the whole of externally observable and measurable (macroscopic) properties of a system, corresponding to visible output of a software system (e.g., loggings). The microstate depicts the whole of microscopic properties of the constituent parts of the system, such as binary values representing the correct of erroneous outcome of a task (i.e., a unit of processing of which we are interested in independent information about whether it has been executed properly). The higher the multiplicity, the more difficult it becomes to identify the precise origin of an observed error. This approach requires a run-time view of the system [7]. To design information systems exhibiting low entropy, two NS theorems have been confirmed, while two additional theorems were proposed as well [7].

This entropy viewpoint can be applied to business processes as well [11], [12]. Again, a business process is considered to be a flow (i.e., including sequences, selections and iterations) of tasks which perform actions on one or more information objects. Considering their execution allows us to define macrostates and microstates on this level as well. The individual values of, for example, the throughput times of all task instantiations correspond to a microstate. The macrostate of a business process is the (aggregated) information available for an observer (e.g., the total troughput time). Multiple microstate configurations consistent with one macrostate (i.e., multiplicity > 1), makes entropy (and the experienced complexity during diagnostics) increase, and typical management questions more difficult to answer. For instance, it becomes unclear which task or tasks in the business process was (were) responsible for the extremely slow (fast) completion (of this particular instance) ofthe business process

No specific guidelines on how to reduce entropy on this level have been formulated yet. Similar to the software level, it is hypothesized that guidelines to achieve stable business processes might reduce entropy as well. As a first step, we assess in this paper the entropy-reducing capability of the first nine available guidelines of Van Nuffel [6]. More specifically, we investigate whether a violation of each guideline increases the multiplicity (and hence, entropy) of business processes.

III. Comparison of guidelines rationales

In this section, we will systematically investigate the first 9 guidelines as proposed by the work of Van Nuffel [6]. For each guideline, we will first provide a brief description. Next, we explore whether not adhering to this guideline would imply an increase in entropy as we defined it earlier. Guidelines of which violations result in additional entropy are then considered to be suitable for entropy control as well.

The first guideline, "**Elementary Business Process**", requires that a business process should be operating on *one and only one* Information Life Cycle Object (ILCO) [6, p. 107]. Not adhering to this guideline would imply a design in which a business process could be operating on multiple ILCOs. For instance, consider both invoicing and manufacturing steps which are mixed up and interacting in one process, and a problem with the total throughput time of finishing invoices is present. At least two situations in which multiplicity > 1 (and entropy arises), can now occur. First, as the business process is concerned with operations on multiple ILCOs, the problematic throughput time of the invoicing steps can be "compensated" by "normal" throughput times of the manufacturing steps. Consequently, the problematic total throughput time of the invoicing activities would not necessarily raise an "alert", even after for instance hypothesis testing on the overall observed mean versus expected mean. Therefore, multiplicity > 1 (and entropy increases): the status reflected by the macrostate (e.g., no problems are reported ("OK")), is conform to multiple microstates (e.g., both "OK" or "Not OK" for the throughput time of the invoicing steps). Further, not demanding that business processes operate on a single information object, also implies that multiple business processes can be operating (unconsciously) on identical information objects (i.e., duplication and copy/paste might occur). Therefore, chances that the problematic total throughput time of the invoicing activities would raise an "alert" become even smaller, as the information on this concern is not properly separated. This situation correlates with our (reduced) observability interpretation of entropy as pointed out in Section II. Second, in case a problem is observed (i.e., the macrostate signals "Not OK"), multiplicity > 1 as well. Indeed, the macrostate conforms to multiple microstates: the "Not OK" result of the total throughput time might be related to the manufacturing steps, the invoicing steps or both. In order to diagnose the problem unambiguously, the process owner should disentangle all steps in the business process, determine the ILCO they belong to, and analyze to which ILCO the overall problem is actually related. Further, we already noted that not demanding a business process to operate on a single information object might result in multiple business processes operating (unconsciously) on identical information objects (i.e., duplication and copy/paste might occur). If the macrostate of multiple business processes (each implementing (duplicate) invoicing steps) goes to "Not OK", chances of identifying "the invoice" as the problematic concern become even smaller, as the information on this issue is not properly separated. This situation correlates with our (reduced) diagnostability interpretation of entropy as pointed out in Section II. Based on these two situations, we can conclude that not adhering to this guideline implies an increased amount of entropy in the business process instantiation space. Therefore, we state that the guideline is suitable for entropy control as well.

The second guideline, "**Elementary Life Cycle Information Object**", defines a *LCIO as an information object not exhibiting state transparency* [6, p. 114]. Combined with guideline 1 this implies that a business process is related to one information object not exhibiting state transparency. In this context, an information object is considered state transparent if it adheres to the NS Separation of States principle and the object has no proper state transitions which should be made explicit [6, p. 118]. Not adhering to this guideline would imply two possible situations: (1) the identification of an information object as a LCIO when it already exhibits state transparency, or (2) not recognizing a non-state transparent information object as a LCIO. Regarding the first situation, the creation of an additional LCIO (and a corresponding business process) for an information object of which the states are already fully reflected by another LCIO, does neither increase of decrease entropy. Indeed, no additional information regarding the microstate configuration is retained or lost (the information regarding the states of one particular LCIO instance is simply duplicated) by identifying this additional LCIO. Regarding the second situation however, an information object not exhibiting state transparency which does not get recognized as a LCIO, will generate an increase in the degree of entropy (i.e., multiplicity > 1). Indeed, as in such case no state transparency regarding the concerning information object is attained, information about its state transitions (and hence, the microstate configuration) is lost. Expressed differently, a multiplicity > 1 will arise during and after execution-time as the macroscopic observations regarding this information object cannot be traced to individual tasks represented by states (i.e., a myriad of microstates are possible). This situation correlates with both our (reduced) observability and diagnostability interpretations of entropy as pointed out in Section II. Consequently, this guideline is not strictly necessary to control entropy in the context of the first situation: theoretically speaking, a state transparent information object can be identified as a LCIO without increasing entropy (albeit without any thinkable benefit). However, the second situation shows that not adhering to this guideline can imply an increased amount of entropy in the business process instantiation space when a non-transparent information object is not recognized as a LCIO. Therefore, we state that the guideline is largely suitable for entropy control and advice its application for this purpose as well. We would further like to add that this guideline actually quite nicely illustrates the core reasoning of designing business processes based on the entropy rationale: for every task of which separate information might be valuable (constituting a so-called "information unit"), a separate state should be defined and related to the information object it is operating on. Therefore, each information object not exhibiting state transparency should be considered as a LCIO, thereby storing information of each individual task performed on it, at its most fine-grained level.

The third guideline, "**Aggregated Business Process**", states that in order to represent an aggregated business process, an aggregated LCIO has to be introduced (p. 121). This guideline relates to the fact that certain aggregated business processes might be necessary to several reasons. First, the orchestration of different business processes (each operating on a single LCIO) by a distinct business process might be necessary. For instance, consider an Order-to-Cash process in which several sub-processes —such as "order entry process", "procurement process", "production process", etcetera— are each individually and successively called, waiting for completion, upon which the next (set of) sub-process(es) is called, completed, etcetera. Second, different (both internal or external) stakeholders might require different perspectives (such as aggregations) due to, for instance, their own functional domain. For instance, in case of very complex business processes, one can imagine that clients or certain actors at a higher management level might be primarily interested in the mere "milestones" (e.g., "order received", "order produced", "order shipped") of a business process, instead of the possible hundreds of more fine-grained states the product might be in during its lifecycle. The guideline under consideration prescribes that such *aggregated processes may only be introduced for orchestrating purposes and in case the business processes under consideration are not able to be designed solely based on guidelines 1 and 2*. Once more, not adhering to this guideline would imply two possible situations: (1) designing an aggregated business process while a redesign based on guidelines 1 and 2 would be possible, or (2) not recognizing a business process for orchestrating purposes while a redesign based on guidelines 1 and 2 is not possible. The first situation would clearly imply an unnecessary combination of two concerns and therefore a violation of guidelines 1 and 2 (as a redesign based on them is still possible). Given the fact that both guidelines were proven to mostly result in an increase of entropy when not adhered to, this situation would equally result in an increase of entropy. The second situation would lead to not recognizing a "combined concern": while each of the underlying concerns have their own LCIO and corresponding business process, the orchestration or "interfacing" between them might constitute a genuine concern as well. This orchestration might entail a relevant information unit and therefore necessary to keep track of when one's aim is to minimize entropy. Imagine an Order-to-Cash process tracking the Order Entry Process, (possibly multiple) Procurement Processes, Production Processes, Delivery Processes, etcetera. While each of these processes clearly designate their own LCIO and therefore, business process, the orchestration between them is crucial to be monitored as well. Indeed, tracking interfacing issues in this Order-to-Cash Process constitutes relevant information (macroscopically) and in case a customer complains about a lately delivered order (i.e., the macrostate), the specific business process (instance) which is causing this delay (Order Entry, Procurement, etcetera) should be identifiable (i.e., the specific microstate) Not identifying the necessary aggregated process would therefore lead to multiple microstates consistent with one macrostate. This situation correlates with both our (reduced) observability and diagnostability interpretations of entropy as pointed out in Section II. We can therefore conclude that not adhering to this guideline implies an increased amount of entropy in the business process instantiation space and state that the guideline is suitable for entropy control as well.

Guideline 4, "**Aggregation Level**", requires that *tasks performed on a different aggregation level should denote a separate business process* (p. 124). An "aggregation level" in this particular guideline is mainly to be understood as focusing on the multiplicities of different information objects (i.e., the different perceived aggregations). For instance, a typical Order within a company might be conceived as being associated with several Product processes, where this Product process at its turn might then again be associated with multiple Part processes. Not adhering to this guideline would imply that it is possible for a business process to execute sequences of tasks situated at different "aggregation levels". Suppose one business process performing a sequence of tasks on a "parent" information object (e.g., "Product") and sequences of tasks on its "child" information objects (e.g., different "Part" instances). As one could argue that such business process is operating on multiple LCIOs, our first two arguments are highly parallel to those of guideline 1. First, such business process design would not guarantee that systematic problems regarding, for instance, the overall throughput time of the sequence of tasks performed on the "child" information object are observed. Indeed, they might become "compensated" by "normal" throughput times of the other tasks, therefore not necessarily raising an "alert" to the observer. Hence, multiplicity > 1 (and entropy increases): multiple microstates ("throughput times OK" and "throughput times Not OK") are consistent with one macrostate ("no problems are reported"). This situation correlates with our (reduced) observability interpretation of entropy as pointed out in Section II. Second, in case a problem is observed (i.e., the macrostate signals "Not OK"), multiplicity > 1 as well. Indeed, the macrostate conforms to multiple microstates: the "Not OK" result of the overall process might be related to the sequence of tasks performed on the "parent" information object, the "child" information object or both. This situation correlates with our (reduced) diagnostability interpretation of entropy as pointed out in Section II. Third, no instance traceability regarding the multiple processed Parts within the single business process seems feasible in such design. Therefore, the same states regarding the "child" information object sequence are activated several times during the execution of the business process. This makes adequate state-tracking (cf. guideline 2) impossible. As a result, the business process owner cannot make the distinction between situations in which the problematic throughput time might be associated with all Part instances in general (i.e., a "systematic" recurring problem) or with one Part instance in particular (and in such case, which specific Product instance). Also in this third situation, this implies multiplicity > 1: one macrostate (i.e., a problem is observed) is consistent with multiple microstate (i.e., the problem is due to Part instance 1, or 2, $\ldots$, or all Part instances): certain parts of the microstate configuration are simply not captured during process execution. Based on these two situations, we can conclude that not adhering to this guideline implies an increased amount of entropy in the business process instantiation space. Therefore, we state that the guideline is suitable for entropy control as well.

Guideline 5, "**Value Chain Phase**", states that the *follow-up of an organizational artifact resulting from a value chain phase should denote a different business process* (p. 132). A value chain phase refers to the rather generic, often recurring structure and parts within aggregated business processes in manufacturing organizations (e.g., Order Entry, Procurement, Production, etcetera), such as for instance described by the SCOR reference model. Not adhering to the above described guideline could lead to the following two situations: (1) the steps related to these value chains are incorporated into the aggregated (i.e., orchestrating) business process, or (2) no more grained steps related to each of these value chain phases are discerned and no states regarding them is kept. In the first situation, this would imply a violation of guidelines 1 as multiple LCIOs (e.g., Order Entry, Procurement, Procurement) are combined into one business process. Further, guideline 4 would be violated as well because most often, these value chain phases have one-to-many or many-to-many relations. Indeed, a Customer Order can typically be related to multiple Purchase Orders and/or Production Orders. The second situation would imply violations regarding guidelines 2 (i.e., no LCIO is identified for several non-state transparent information objects) and 3 (i.e., an aggregated business process is designed when there are still some opportunities for redesign based on guidelines 1 and 2). A situation in which no relevant states regarding the tasks constituting a value chain phase should be identified, seems rather unlikely as this would allow to model almost all necessary activities of a typical manufacturing company within one business process having 5 to 8 tasks. Consequently, as we should earlier how violations regarding guidelines 1 to 4 result in multiple microstates consistent with one macrostate, we can conclude that violating this guideline would generate a multiplicity > 1 as well. Therefore, we state that the guideline is suitable for entropy control as well.

Guideline 6, "**Attribute Update Request**", states that *a task sequence to update an attribute of a particular LCIO that is not part of its business process scenarios, is represented by an Attribute Update Request business process* (p. 135). This guideline is subject to two specific conditions. First, it has to concern an update operation for which one single functional task is not sufficient to complete the update request, but rather a sequence (i.e., "process") of activities is required. Second, it concerns update requests which are not part of a branch within the regular business process scenarios. Consequently such procedures can be instantiated several times and during several different "states" of the lifecycle of the information object regarding which the update request is actually aimed at. Additionally, such process (verifying for instance the validity of updating a certain information object attribute with a certain new value) will typically differ for each individual attribute. Not adhering to this guideline would imply that tasks for handling an attribute update request, not part of the regular business process scenario, becomes incorporated into the flow of the LCIO of which the attribute is requested to be update. Again, such situation can be seen as a violation regarding several of the above mentioned guidelines. Indeed, not separating such task sequences would lead to a business process operating on multiple ILCOs and —at the same time— one concern being dispersed over several places within one business process (i.e., all the life cycle states in which the update request is allowed), thereby violating guideline 1. Second, the design would make the proper tracking of states impossible as at any point of the business process execution (thereby indirectly violating guideline 2) as each time an update request is initiated, the state of the regular business process is suddenly (possibly repeatedly) changed to states regarding this update request. Third, as attribute update requests can be performed several times during one instance of the "parent" business process, both concerns relate in a one-to-many multiplicity, thereby violating guideline 4. Consequently, as we showed earlier how violations regarding guidelines 1, 2 and 4 result in multiple microstates consistent with one macrostate, we can conclude that violating this guideline would generate a multiplicity $>$ 1 as well. Therefore, we state that the guideline is suitable for entropy control as well. Indeed, from an organization diagnostics (i.e., entropy) viewpoint, it clearly makes sense to separate such sequence of tasks for future reference. For instance, the calculation of certain measures and the solution for certain managerial questions such as: "how often are such requests accepted/denied and for which reason" or "can we see any relation between the outcome of the update requests and its input values" are only able to be solved in an efficient way when this task sequence is properly separated in its own business process module and not unconsciously repeated in other places throughout the business process repository.

Guideline 7, **Actor Business Process Responsibility**, states that *tasks, of which the task allocation genuinely belongs to a different business process owner, should be designed into a separate business process* (p. 139). This guideline only applies in very stringent cases. For example, in case legislation or internal audit rules prescribe that different owners should be responsible for other (parts of) task sequences, this guideline applies. Mostly, the guideline is applicable when different parts of a task sequence are performed by different organizations. In such cases, the respective task allocations are logically situated at one of these different organizations as well. From an entropy viewpoint, let us consider the case in which the mentioned guideline is not adhered to. In such case, a business process could consist of a combination tasks which belong to genuinely different business process owners. Each task still has an attribute regarding which actor is allowed or required to perform the task. However, no information is available regarding who is doing the task allocation (e.g., the manager of organization who determines who is doing what). If such information should be retained, the appropriate level seems to be the business process level, as it concerns a sequence of multiple tasks. In case this information is relevant but however no distinct business process would be designed, a multiplicity > 1 (and hence, entropy) arises as one macrostate (e.g., a problem regarding the overall process) complies with multiple microstates (was the task allocation responsibility situated at person A, B, or C?). This situation correlates with our (reduced) diagnostability interpretation of entropy as pointed out in Section II. Therefore, in case the information regarding task allocation responsibility is relevant, a different business process should be identified from an entropy viewpoint to allow for this task allocation responsibility to be traceable. Indeed, this guideline calls to create an additional level of "process responsibility" (i.e., who allocates tasks among different actors and takes responsibility that they are carried out adequately), in addition to the responsibility for one or multiple tasks. Therefore, we state that the guideline might be suitable for entropy control as well. However, in line with the work of Van Nuffel [6] we stress that identifying additional business processes based on this guideline should be done with extreme precaution to avoid unnecessary additional business processes and, hence, only in cases where a different task allocation responsibility is relevant for diagnostability purposes.

Guidelines 8 and 9 as proposed by Van Nuffel [6], propose two specific business process types to be identified. Guideline 8, "**Notifying Stakeholders**" states that the *communication of a message to stakeholders (in the correct format, incorporating fault handling, etcetera) constitutes a distinct business process* (p. 143). Guideline 9, "**Payment**" states that the *payment of a particular amount of money to a particular beneficiary should equally constitute a distinct business process* (p. 146). Not recognizing these two concerns as distinct business processes could again create two possible situations: (1) integrating the tasks for the notification and payment in other business processes or (2) not specifying their constituting tasks at all. It is clear that the first situation would violate guideline 1 (multiple ILCOs operating within one business process) and 4 (for example, multiple notifications can be sent within the scope of one "parent" business process instantiation). The second situation would violate guideline 2 as a non-state transparent information object is not identified as a separate LCIO. Consequently, as we showed earlier how violations regarding guidelines 1, 2 and 4 result in multiple microstates consistent with one macrostate, we can conclude that violating this guideline would generate a multiplicity > 1 as well. Therefore, we state that guideline 8 and 9 are suitable for entropy control as well. Obviously, designing these task sequences as separate business processes is useful from an organizational diagnostics (i.e., entropy) viewpoint as. Indeed, both the payment of a particular amount in a particular format to a particular beneficiary at the right time, as well as communicating a certain message in a particular format at the right time while maintaining integrity, are often recurring functionalities within typical business processes. As a consequence, due to their frequently occurring nature, a business process owner would typically be interested in certain characteristics of each of these separately recurring tasks sequences: how long do they take to execute, how many times do they result in an error, etcetera. Focusing on these aspects might generate considerable efficiency gains as, for instance, improving the quality metrics or throughput time of the payment process with 5% might entail huge organizational effects as the changes are "expanded" throughout the whole organization. However, these analyses and improvements can only be performed when "payments" and "notifications" are designed into separate business processes. Otherwise, systematic problems regarding one of the concerns might not be noticed (cf. the observability issue of Section II) or might not be unambiguously traced to the right concern (cf. the diagnostability issue of Section II)

IV. Discussion, limitations and future research

This paper aims to contribute to our research line on how to prescriptively design business processes regarding certain criteria (such as low complexity and high evolvability). In earlier work, a set of prescriptive guidelines has been proposed from the stability perspective, and the applicability of entropy to study process complexity has been reported. The main focus in this paper was to verify whether the already existing guidelines from the stability viewpoint align with this entropy reasoning [12], [11]. Due to page limitations, we were only able to investigate a small subset of the guidelines of Van Nuffel for this purpose [6]. We found that most of the investigated guidelines are rather consistent among both approaches: guidelines required to attain evolvability seem to enable low complexity and vice versa. A small exception was noticed for guidelines 2 and 7. Regarding the former, it was observed that —theoretically— entropy does not increase when a state transparent information object is identified as a LCIO. Regarding the latter, it was argued that the application of the specific guideline should be performed even more thoughtfully and exceptionally from an entropy viewpoint as its necessity in many situation seems not really compelling.

This consistency might seem surprising, since the evolvability analysis focuses on the mere *design-time* of business processes, which means that the harmful effects its aims to resolve (the so-called "combinatorial effects") are situated on this perspective: a functional change which causes N changes in the business process design. In contrast, the complexity analysis focuses on avoiding harmful effects during *execution-time*: a multiplicity > 1 (which we could coin as an "uncertainty effect") only manifests itself when the business processes are executed. While these effects are caused by choices made at design-time, this distinction illustrates the need for more insight at the execution-time of business processes. Current business process modeling notations (e.g., BPMN) focus primarily on design-time models. Moreover, the criteria both approaches use to delineate and identify the different business processes and their constituting tasks, differ. The evolvability approach employs the concept of "*change drivers*" (i.e., parts within the business process design which are assumed to change independently) to identify and isolate concerns, whereas the complexity approach employs the concept of "*information units*" (i.e., these parts within the business process design of which independently traceable information is assumed to be needed later on). Since most of the stability-related guidelines largely align with our entropy reasoning, we might conclude that the concerns which should be used to delineate and identify business processes or tasks are determined by the union of "change drivers" and "information units". Given the additional, more in-depth analysis of the entropy approach by incorporating the execution-time perspective (e.g., the importance of traceability), additional concerns which do not seem to be necessary from the evolvability perspective, might indeed be potentially identified in future research. Moreover, this preliminary analysis is limited to the first nine guidelines of Van Nuffel [6], and future research should elaborate on the consistency of other guidelines.

Notwithstanding the limitations and need for future research, this paper can claim a number of contributions. First, we further contributed to the enterprise and business process engineering field by elaborating on the usefulness to take an entropy perspective for studying the complexity of business processes. Second, we validated the suitability of a set of (already existing) business process design guidelines in this context as a first step towards a Design Theory [13]. In literature, it is generally acknowledged and even encouraged that such design efforts are guided by principles from related scientific fields (i.e., "kernel theories") [14], such as the concept of entropy from thermodynamics. Third, Design Science research acknowledges logical reasoning as one the possible evaluation methods in design science [15]. Therefore, next to our efforts performed in earlier work, this paper constitutes an additional validation base for the applicability of (a part of) the guidelines of Van Nuffel [6].

V. Conclusion

Contemporary organizations need to be agile regarding both their IT systems and organizational structures (such as business processes). Normalized Systems theory has recently proposed an approach to build evolvable IT systems, based on the systems theoretic concept of stability. However, its applicability to the organizational level, including business processes, has proven to be relevant in the past and resulted a.o. in a set of 25 guidelines for designing business processes. This paper investigated the validity of 9 of these guidelines from another theoretical perspective, more specifically, entropy from thermodynamics. We concluded that the investigated guidelines are rather consistent among both approaches: guidelines required to attain evolvability seem to enable low complexity (i.e., entropy) and vice versa. However, future research is definitely needed in this domain: for instance, 14 guidelines are still to be investigated and additional guidelines might potentially be investigated from the entropy perspective as well.

Acknowledgment

P.D.B. is supported by a Research Grant of the Agency for Innovation by Science and Technology in Flanders (IWT).

References

[1] E. Overby, A. Bharadwaj, and V. Sambamurthy, "Enterprise agility and the enabling role of information technology," *European Journal of Information Systems*, vol. 15, no. 2, pp. 120–131, 2006.

[2] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, "Seven process modeling guidelines (7pmg)," *Inf. Softw. Technol.*, vol. 52, no. 2, pp. 127–136, Feb. 2010.

[3] L. Brehm, A. Heinzl, and M. Markus, "Tailoring erp systems: a spectrum of choices and their implications," in *Proceedings of the 34th Annual Hawaii International Conference on System Sciences*, 2001.

[4] H. Mannaert, J. Verelst, and K. Ven, "The transformation of requirements into software primitives: Studying evolvability based on systems theoretic stability," *Science of Computer Programming*, vol. 76, no. 12, pp. 1210–1222, 2011.

[5] ——, "Towards evolvable software architectures based on systems theoretic stability," *Software: Practice and Experience*, vol. 42, no. 1, pp. 89–116, January 2012.

[6] D. Van Nuffel, "Towards designing modular and evolvable business processes," Ph.D. dissertation, University of Antwerp, 2011.

[7] H. Mannaert, P. De Bruyn, and J. Verelst, "Exploring entropy in software systems : towards a precise definition and design rules," in *The Seventh International Conference of Software Engineering Advances (ICSEA)*, 2012, pp. 84–89.

[8] S. P. Regev, G. and R. Schmidt, "Taxonomy of flexibility in business processes," in *Proceedings of the 7th Workshop on Business Process Modelling, Development and Support (BPMDS'06)*, 2006.

[9] M. R. R. N. M. N. Schonenberg, H. and W. van der Aalst, "Process flexibility: A survey of contemporary approaches," in *Advances in Enterprise Engineering I*, 2008, pp. 16–30.

[10] L. Boltzmann, *Lectures on gas theory*. Dover Publications, 1995.

[11] P. De Bruyn, P. Huysmans, H. Mannaert, and J. Verelst, "Understanding entropy generation during the execution of business process instantiations: An illustration from cost accounting," in *Advances in Enterprise Engineering VII*, ser. Lecture Notes in Business Information Processing, H. Proper, D. Aveiro, and K. Gaaloul, Eds. Springer Berlin Heidelberg, 2013, vol. 146, pp. 103–117.

[12] P. De Bruyn, P. Huysmans, G. Oorts, and H. Mannaert, "On the applicability of the notion of entropy for business process analysis," in *Proceedings of the second international symposium on Business Modeling and Software Desgin (BMSD2012)*, B. Shishkov, Ed., 2012, pp. 128–137.

[13] S. Gregor and D. Jones, "The anatomy of a design theory," *Journal of the Association for Information Systems*, vol. 8, no. 5, pp. 312–335, 2007.

[14] J. Walls, G. Widmeyer, and O. El Saway, "Building an information system design theory for vigilant eis," *Information Systems Research*, vol. 3, no. 1, pp. 36–59, 1992.

[15] F. Müller-Wienbergen, O. Müller, S. Seidel, and J. Becker, "Leaving the beaten tracks in creative work - a design theory for systems that support convergent and divergent thinking," *Journal of the Association for Information Systems*, vol. 12, no. 11, pp. 714–740, 2011.

Towards Ontology-Driven Approach for Data Warehouse Analysis

Case study : Healthcare domain

Lama El Sarraj[1,2], Bernard Espinasse[1]
[1]LSIS UMR 7296
Université d'Aix-Marseille,
Marseille, France
{firstname.lastname}@lsis.org

Thérèse Libourel[3]
[3]Espace-Dev UMR 228
Université Montpellier 2
Montpellier, France
therese.libourel@univ-montp2.fr

Sophie Rodier[2,]
[2]Assistance publique–Hôpitaux Marseille
DSIO
Marseille, France
{firstname.lastname}@AP-HM.fr

***Abstract*—Understanding, reusing, and maintaining data warehouse resources is a key challenge for data warehouse users. Data warehouses resources are shared by different groups of users. The interpretation of information is subjective, it depends on user knowledge. Thus, a resource, like a data cube, is interpreted differently from a user to another. Unfortunately, misinterpreting data could induce serious problems and conflicts. To guarantee homogenous interpretation of data warehouse resources additional information is necessary. To tackle these challenges we propose to use ontologies to help the users in the exploitation of data warehouses. In this paper we propose an ontology-driven approach that represents data warehouse, dimensions and facts semantically enriched by their equivalent domain concepts and related to final resources provided by this data warehouse.**

Keywords- data warehouse; ontology; decision information systems; decision making; healthcare institution management

I. INTRODUCTION

Several surveys proved that big companies need efficient Decision support systems (DSS) and seek to expand the number of users over their DSS. To that aim, researchers found that companies need to have flexible decision tools, especially with, users' requirements and domain resources. A DSS is a collection of many tools or applications; we call them in this paper resources; that enable users to analyze, to query and to visualize a huge volume of data. In general, those data are stored in a data warehouse, and a set of Business Intelligence (BI) tools dedicated for data treatment and helping users (directors, managers, analysts, etc.) to make decisions.

Data Warehouse (DW) is the center of the DSS. DW is « a subject oriented, nonvolatile, integrated, time variant collection of data in support of management's decisions» [1]. In this paper we only consider resources provided by a data warehouse in a decision support system. To facilitate the task of DW analysis and treatment, a subset of the DW is created, it is called data mart. A data mart is oriented to a specific business need or a particular user requirement. Most of the times, data mart are organized in a multidimensional structure [2]. Data are represented like a point in a multidimensional space, visualized like a data cube (see Fig.1) [3]. They give users the possibility to synthetize and analyze data from three (or higher) dimensional array of values and various granularity levels. To manipulate data provided by the DW, end-users could use On Line Analytical Processing (OLAP) techniques, classic techniques, or even dashboards.

Taking user requirements into account is very important for the success or the failure of the DW [4], especially when users belong to different domains. The exploitation level of DW, as well as the preliminary conception level, is mainly based and adapted to user requirements [5]. Most research works devoted for DW focus on the approach design [6], [7], [8]. Even if these approaches are successful at the conceptual level knowledge about the data warehouse resources is still needed. It is important that users understand the semantic around the information he analyses and have a visibility about other resources that could help them to make efficient analysis.

The goal of this work is to design an ontology that relates data warehouse structure, resources and domain concepts. In consequence, in this paper we address two research questions:

- What are the competencies questions that our ontology takes in consideration?
- What are the concepts that compose the ontology to help decision makers in their analysis to understand indicators provided from a data warehouse?

Our research is supported by the public hospitals of Marseille; Assistance Publique Hôpitaux de Marseille (APHM). In this context we will present a case study from the healthcare domain specific to financial program based on the Program of Medicalization of Information Systems (PMSI) common to all French healthcare institutions.

This paper presents a new ontology-driven approach for DW personalization to resolute the semantic problematic related to the heterogeneous domains we applied our approach in healthcare management domain. The paper is organized as follow. Section II presents a case study from the healthcare domain. Section III presents the competencies questions that give an idea about the possible scenarios possible to help users in his analysis. Section IV presents the needed background. Section V presents an ontology-driven approach. Section VI presents an ontology-driven framework. Finally, before we conclude we present in section VII the related works.

II. CASE STUDY

In this section we will present a case study from the healthcare domain specifically applied in the Program of Medicalization of Information Systems (PMSI). This case study is a good example that represents heterogeneous users that share same data warehouse.

In the French healthcare management system the PMSI has a central place. PMSI is a French adoption for the concept of Professor R. Fetter (Yale university, United States of America) to finance hospitals. PMSI specify the cost of sojourn based on diagnosis related groups that classes the hospitalization of patients in homogeneous and coherent medico-economic groups. This concept is applied in several countries like United States of America, England, etc.

In the healthcare domain users belong to the medical domain (doctors, pharmacists, biologists, etc.) whereas others don't (financial affaire managers, computer scientists, human resources, etc.). We should note that our approach is not limited to the healthcare domain. It could be applied in other business contexts where users are from different domains. This is, in general, the case of big institutions.

In this context we will take the example of a data warehouse. Fig.1 represents a data warehouse conceptual model for "PMSI activity" analysis. This DW conceptual model is composed of a fact table, dimensions, and measures.

Fact table = {Activity_PMSI}
Dimensions = {Date, Structure, Age, Exit_Mode, International_classification_of_desieases, Diagnosis_related_groups }
Measures = {Number of patient, …}

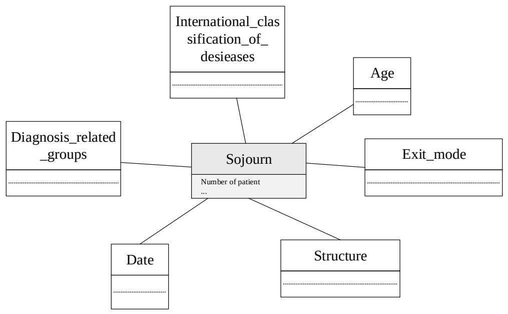
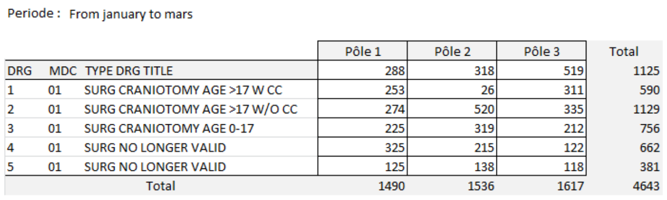

International_clas sification_of_ desieases
Age
Diagnosis_related _groups
Sojourn
Number of patient
...
Exit_mode
Date
Structure

Figure 1. PMSI activity data warehouse conceptual model.

The multidimensional table (MT), MT = (M, D), where M is a set of measure and D is a set of dimensions. We will take an example of a multidimensional pivot table, presented in Fig. 2, for ethics reason we have taken fictive data:

D_1 = "Structure " (dimension level "pôle")
D_2 = "Diagnosis Related Groups" (attributes: DRG, MCD, TYPE DRG TITLE)
M_1 = "number of patients" (calculated measures: total of M1 per Diagnosis Related Groups, total of M1 per pole, total of M1 for all DRG and poles.



Periode : From january to mars

			Pôle 1	Pôle 2	Pôle 3	Total
DRG	MDC	TYPE DRG TITLE	288	318	519	1125
1	01	SURG CRANIOTOMY AGE >17 W CC	253	26	311	590
2	01	SURG CRANIOTOMY AGE >17 W/O CC	274	520	335	1129
3	01	SURG CRANIOTOMY AGE 0-17	225	319	212	756
4	01	SURG NO LONGER VALID	325	215	122	662
5	01	SURG NO LONGER VALID	125	138	118	381
		Total	1490	1536	1617	4643

Figure 2. PMSI pivot table.

In this research work we will take into consideration resources based on data warehouses sources and that represent data in a multidimensional table (defined by of measure, an operations on the measure, two or three dimensions, and a filter). In this context we noticed many difficulties:

Semantic lack

Users don't interpret the results in the same way. They need information about:

- Data warehouse concepts: dimensions definition, measures calculation methods and their sources
- Requirements expression heterogeneity: users don't belong to the same domain. They don't express their need with the same terms. For example: number of sojourn could be expressed as number of venue

Analysis needs

Most of the times, users need to analyze many resources to take a decision. In big institutions the big number of resources makes this task complicated. To facilitate this task, users need a global vision about the existing analysis axes. Thus, users need to have a global vision about the data warehouse structure to visualize the possibilities or existing resources that could help him to take a decision.

Finally, these difficulties lead us to think about a new semantic approach that structure the concepts related to the data warehouse based on ontologies.

III. COMPETENCIES QUESTION

In this section we exemplify and define possible scenarios to interrogate our ontology.

Entry 1: Data warehouse concept.

Output:

1. *Related data warehouse concept* -- Measures analysis -- What are the different measures related to an analysis axe? What is the different analysis axes related to a measure?
   Dimensions (Analysis axes) -- What are the measures that could be analyzed over a dimension?
2. *Resources concept* -- What are the existing resources to analyze a measure?
3. *Domain concepts* -- What are the existing measures to analyze a domain concept?

Entry 2: Resources concept.
Output:

1. *Data warehouse structure concepts* -- Which is the data warehouse (data mart) that provides a resource
2. *Domain concepts* -- What are the existing resources to analyze a domain concept?

Entry 3: Domain concept.
Output:

1. *Data warehouse structure* -- Which is the data warehouse (data mart) related to this domain concept?
2. *Resources concept* -- What are the resources to analyze a domain concept?

Those scenarios could be treated by using ontology technologies to visualize and have semantic to facilitate the analysis.

IV. BACKGROUND

In this section we will define the ontology and present some researches that have used ontology for the multidimensional systems.

A. *Ontologies*

Ontology is an explicit specification of shared conceptualization [9]. Different ontologies are proposed to define ontologies. W3C consortium recommends Ontology Web Language (OWL) to define ontologies. This language is based on the description Logic (DL) [10], it gives the opportunity to reason and represent structured knowledge. The DL language represents knowledge with concepts and roles. The concepts described as a set of individuals (instances) and roles describing a binary relation between individuals.

A knowledge base is represented with an ABOX (assertion box) and a TBOX (terminological box). An ABOX represent extensional knowledge (instances), TBOX describes the intentional knowledge of the domain as axioms.

We present the ontology with 4-uplet <C, P, ClassPropt, ClassAssoc> that concerns the TBOX.

Our ontology describes concepts to relate domain, resources and data warehouse structure. We consider:

- C represents the classes of the ontological model
- P represents the properties of the ontological model. P is partitioned into :
  - P_{value} : represents the characteristics properties
  - P_{fct} : represents domain dependent properties
- ClassPropt : C -> 2P relates each class to its property
- ClassAssoc : C -> (Opr, Expr (C)) is an expression that associate to each class an operator (inclusion or exclusion) and an expression to other classes.

B. *Multidimensional system*

We consider that DW resources are multidimensional table that represent a slice of the cube. The DW ontology registers the DW conceptual schema and the resources provided from this DW. For other purposes, several researchers like Prat et al [11] represents a multidimensional model with an OWL-DL ontology model, based on description logic [12], and define the transformation rules from the multidimensional level into OWL-DL ontologies. We will use these transformation rules to generate an OWL ontology of the DW model, based on transformations rules proposed in the work of Prat et al [11].

V. ONTOLOGY-DRIVEN APPROACH FOR DATA WAREHOUSE ANALYSIS

In this section we briefly present our approach and the architecture of our system.

Our approach focuses on two key requirements to address the research problem:

- It represents ontology architecture to describe knowledge about decision support system
- It provides an ontology-driven approach to help users in their analysis

A. *Approach architecture*

Our functional architecture Fig. 3 is based on three inter-related concepts, in order:

- Domain concepts
- Data warehouse structure
- Resources

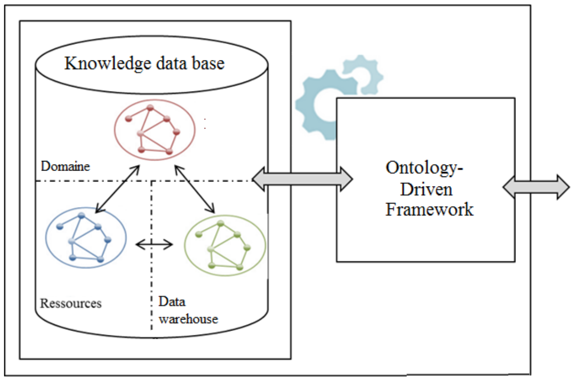


Figure 3. Approach architecture.

The framework system that we propose is based on an ontology interrelating three concepts (domain, DW and resources) to help users in the analysis task.

B. *Ontology concepts*

We will define the three concepts that compose our ontology. These concepts are necessary to help users in the analysis process:

Domain concepts structure: presents concepts of the domain and the relation between them. A decision is based on one or many indicators. In the analysis processes the user check the information's that he already know. However, most of the times user needs additional indicators to make his analysis. The domain description wills provide the information about the relation between domain concepts.

Data warehouse structure: the multidimensional model associated to the data warehouse organizes data into facts and dimension. Facts represent the subject of analysis and dimensions represent the axis of analysis. Fact table is the center of the multidimensional model. It stores elementary indicators, called measures. Dimensions can form hierarchies, structured in different granularity levels.

Resources structure: resources are provided by the data warehouse. Resources regroup information necessary for the analysis. To understand a component information about the indicator are needed like: calculation method, unit of measure, calculation period, date of creation, date of update, date of validity, objective, definition and the relation with the data mart.

C. Ontology connection

To connect those three concepts we will follow four steps:

1. Define domain ontology or use an existing domain ontology
2. Generate the data warehouse structure ontology based on the transformation rules proposed in the work of Prat et al [11].
3. Associate the data warehouse structure to the domain ontology, this step could be accomplished in several methods, for example :
    - Administrator relates data warehouse concepts to the domain concepts
    - Automatically align the data warehouse structure ontology with the existing domain ontology
4. Associate to the data warehouse concepts existing resources Ontology architecture

D. Ontology architechture

We will formalize our ontology by the triple < O_{DW}, O_D, *Map*> where:

- O_D is the domain ontology which provides a schema about the domain
- O_{DW} is a data warehouse schema which describes the resources (DSS components) related to the data warehouse
- Map is the mapping between O_{DW} and O_D which establish the connection between domain concepts and the DSS components

This ontology can be used for many purposes with ontology-based software. In the first hand, to give a vision about the relation between DW, resources and domain concepts, in the other hand, to propose for users other related resources to accomplish his analysis, based on the relation of the three concepts the resources, the data warehouse concepts and the domain concepts. Fig. 4 presents the ontology architecture meta-model to implement the knowledge base of the framework.

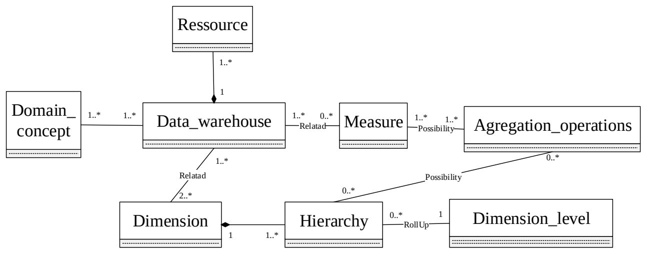


Figure 4. Ontology metamodel.

This ontology model represents the concepts related to the data warehouse. Each data warehouse is composed of zero or many measures and related to two or many dimensions. Hierarchies are composed of one or many dimensions. It is possible to effectuate operations on measures and aggregation according to the dimensions levels.

The proposed ontology model has been designed as follow to give high expressiveness about data warehouse components and to show the relation between DW concepts, resources (DSS components) and domain concepts.

VI. ONTOLOGY-DRIVEN FRAMEWORK

In this section we will present a framework based on our ontology. We implemented an ontology based on healthcare domain. Thus, this semantic structure will help users to discover and retrieve resources related to their domain and their first need.

To test our method we chose to implement OWL ontology with Protégé editor [13], and then we will use protégé to interrogate and visualize ontology with OntoGraph Fig. 5.

A. Methods

To create our OWL ontology we use “Protégé”, an open source Java tool providing an extensible architecture for the creation of customized knowledge-based applications.

1. Create three classes Data_Warehouse, Domain, and resources
2. Export existing domain ontology or create new domain ontology. These ontology concepts will be a subset of the domain class
3. Export data warehouse conceptual model ontology. To pass from the data warehouse conceptual model to OWL we applied the transformations rules proposed by [14]. Data warehouse concepts will be a subset of the Data_Warehouse class
4. Relate the data warehouse concepts to domain concepts. This task can be automatic by using existing ontology mapping tools; in this work we’ll not consider this option. To relate data warehouse concepts to domain concepts ontology administrator will refer to each data warehouse concept the equivalent, opposite, etc. concept in the domain ontology. For example, the data warehouse

dimension "Diagnosis_Related_Groups" will be related to "DRG" class of the domain ontology

5. Relate the resources provided by the data warehouse to their corresponding concepts. For example, the resource named "PMSI_activity" allows user to analyze the PMSI activity per month and per medical unit. So, this resource will be related to Data_Warehouse subclasses dimensions month and medical units

B. Visualization

We will consider the example of the data warehouse presented in the healthcare domain. We will propose an ontology-driven framework.

Input: is a need expressed with a term or a group of terms.

Output: are concepts related to this need, about resources concepts, domain concepts, and data warehouse structure concepts.

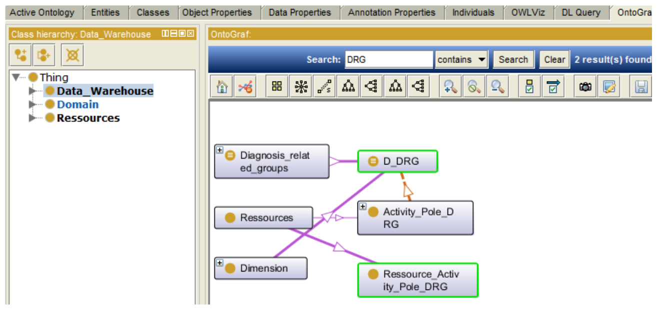


Figure 5. Example, retrieve 'DRG' concept from the ontology.

Thus, the user expresses his need with one or more keywords for example DRG.

- Domain concept: DRG is equivalent to "diagnosis related groups"
- DW concept: DRG is a dimension

So as Fig. 5 shows the resulting visualization of the ontology shows the existing concepts that contains DRG, equivalent and related concepts.

VII. Related Works

In the literature researches in the data warehousing field have already explored the ontology-based data warehouses and the personalization.

In the first hand, in the ontology-based data warehouses field researches are based on the multidimensional schema design, representation and its summarizability.

Prat et al [14] represent a multidimensional model with an OWL-DL ontology model to check the multidimensional model and its summarizability. Niemi and Niinimäki [15] provide an RDF model of an OLAP cube, they focus on the relationship between measure and dimension attributes and its effect on summarizability. They define the concept of measure-dimension consistency and they show how to conclude it from OLAP ontology. The OLAP ontology is constructed with semantic web technologies and is basically used to help users for OLAP cube construction and querying. Nebot et al [16] proposes a framework for designing semantic data warehouses. They propose the Semantic Data Warehouse to be a repository of ontologies and semantically annotated data resources and propose an ontology-driven framework to design multidimensional analysis models for Semantic Data Warehouses.

In the other hand, in the personalization of the data warehouse field we can distinguish three main objectives:

- *Customizing data sources schema* [17], [18] adapting the data structures to a specific needs of users
- *Customizing queries visualization* [19], or representation [20]
- *Recommendation of OLAP queries* [21, 22] to assist in the exploration of the ED

We also find the *personalization of the DW by recommendation that* can be associated to various works such as [17], [21], [23]-[26].

All these personalization techniques are not based on ontologies. Only Jerbi et al [27] adds semantic by annotation of the DW schema but his technique is not based on ontologies.

In our research we use ontology to personalize users need and retrieve not only semantic information about DW or cube schema but also the eventual existing resource like files (PDF, Excel, etc.), OLAP queries, etc. To that aim we integrate domain and resources concepts to our DW ontology.

VIII. Conclusion

The Data Warehouse (DW) resources are shared by users from heterogeneous domains. Those resources could be interpreted differently from a user to another. Consequently, semantic about those resources is necessary to guarantee the coherence of the analysis. Ontologies are effective solutions to add semantic to concepts. They facilitate the management of data, clarify and give a sense to ambiguous concepts.

Ontologies have been adopted by companies. Different solutions are offered to manage and query these data. In this paper we implemented the ontology with Protégé, interrogated and visualized the ontology with OntoGraph.

The study of concepts from healthcare domain confirms the need of semantic to help users in the analysis of resources provided by DW. One of the main characteristic of our proposed ontology architecture is that it provides a connection between domain concepts, data warehouse structure and data warehouse resources, this connection provide semantic information about resources and help users to choose other resources that can help him in his analysis. This personalization task is based on resources related to connected domain concept in the ontology.

Furthermore, the main asset of our proposition is that it combines ontology and data warehouse to add semantic to resources analysis.

We should note that our approach is not restricted to the healthcare domain it could be applied for any domain for the retrieval of data warehouse resources.

This work leads to many other tasks. In future work, tasks that should be considered (i) test the integrity of the ontology when adding new concepts (like new resources), (ii) extension of this approach to add other type of resources and data source provided from decision support system but not related to the data warehouse, (iii) study different scenarios of the ontology evolution, (iv) validate our approach in a larger context.

REFERENCE

[1] W. H. Inmon, Building the data warehouse, New York, NY, USA.: John Wiley & Sons, 1992.

[2] W. Lehner, "Modelling Large Scale OLAP Scenarios," in Advances in Database Technology (EDBT), 1998, pp. 153-167.

[3] A. Bosworth, J. Gray, A. Layman , H. Pirahesh "Data Cube : A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total," Data Min. Knowl. Discov., pp. 152-159, 1995.

[4] S. Rizzi, A. Abello, J. Lechtenborger, J. Trujillo "Research in data warehouse modeling and design: dead or alive?," Proceedings of the 9th ACM international workshop on Data warehousing and OLAP - DOLAP '06, pp. 3-10, 2006.

[5] M. Golfarelli, "From user requirements to conceptual design in data warehouse design – a survey," 2009.

[6] R. Kimball, and M. Ross, The data warehousing toolkit, New York: John Wiley&Sons, 1996.

[7] N. Prat, and J. Akoka, "From UML to ROLAP multidimensional databases using a pivot model," in 8èmes Journées Bases de Données Avancées, 2002, pp. 24.

[8] A. Tsois, N. Karayannidis, and T. K. Sellis, "Mac : Conceptual data modeling for olap," in 3rd International Workshop on Design and Management of Data Warehouses (DMDW 2001), Theodoratos 2001, pp. 5.

[9] T. Gruber, "A translation approach to portable ontology specification," Knowledge Acquisition, vol. 5, no. 2, pp. 199-220, 1993.

[10] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi " The description logic handbook: theory, implementation, and applications," Cambridge University Press 2003.

[11] N. Prat, J. Akoka, and I. Comyn-Wattiau, "Transforming multidimensional models into OWL-DL ontologies," in RCIS, 2011.

[12] B. Grosof, I. Horrocks, R. Volz et al., "Description logic programs: combining logic programs with description logic," in WWW, Budapest, Hungary, 2003.

[13] Stanford Center for Biomedical Informatics Research. 14/08/2013, 2013; http://protege.stanford.edu/.

[14] N. Prat, I. Megdiche, and J. Akoka, "Multidimensional Models Meet the Semantic Web: Defining and Reasoning on OWL-DL Ontologies for OLAP," in DOLAP, Hawaii, USA, 2012.

[15] T. Niemi, and M. Niinimäki, "Ontologies and summarizability in OLAP," in Proc. of SAC'10, Sierre, Switzerland, 2010.

[16] V. Nebot, R. Berlanga, J. Pérez, M. Aramburu, T. Pederson "Multidimensional integrated ontologies: a framework for designing semantic data warehouses," Journal on Data Semantics, vol. XIII, 2009.

[17] F. Bentayeb, O. Boussaid, C. Favre, F. Ravat, O. Teste "Personnalisation dans les entrepôts de données : bilan et perspectives," in Entrepôt de Données et Analyse en ligne (EDA), 2009.

[18] I. Garrigos, J. Pardillo, J.-N. Mazon, J. Trujillo., "A Conceptual Modeling Approach for OLAP Personalization," in Conceptual Modeling-ER Verlag Berlin Heidelberg, 2009, pp. 401-414.

[19] L. Bellatreche, A. Giacometti, P. Marcel, H Mouloudi, D. Laurent "A personalization framework for OLAP queries," in 8th International Workshop on Data Warehousing and OLAP, DOLAP'05, Bremen, Germany, 2005, pp. 9-18.

[20] D. Xin, J. Han, H. Cheng, X.,A. Li, "Answering top-k queries with multi-dimensional selections: The ranking cube approach," in VLDB, 2006, pp. 463-475.

[21] A. Giacometti, P. Marcel, and E. Negre, "A Framework for Recommending OLAP Queries." pp. 73-80.

[22] A. Giacometti, P. Marcel, and E. Negre, "Recommending Multidimensional Queries " in DaWaK, 2009, pp. 453-466.

[23] C. Sapia, "On Modeling and Predicting Query Behavior in OLAP Systems," in DMDW, 1999, pp. 2.1-2.10.

[24] G. Chatzopoulou, M. Eirinaki, and N. Polyzotis, "Query Recommendations for Interactive Database Exploration," in SSDBM, 2009, pp. 3-18.

[25] H. Jerbi, F. Ravat, O. Teste, G. Zurfluh, "Applying Recommendation Technology in OLAP Systems " in ICEIS, 2009, pp. 220-233.

[26] A. Giacometti, P. Marcel, E. Negre, A. Soulet, "Query recommendations for OLAP discovery driven analysis." pp. 81-88.

[27] H. Jerbi, F. Ravat, O. Teste, G. Zurfluh, "Management of Context-Aware Preferences in Multidimensional Databases. ," in ICDIM 2008, pp. 669-675.

Light-PubSubHubbub: A Lightweight Adaptation of the PubSubHubbub Protocol

Porfírio Dantas, Jorge Pereira, Everton Cavalcante, Gustavo Alves, Thais Batista
DIMAp – Department of Informatics and Applied Mathematics
UFRN – Federal University of Rio Grande do Norte
Natal, Brazil
{enghaw13, jorgepereirasb}@gmail.com, {evertonrsc, gustavo}@ppgsc.ufrn.br, thais@ufrnet.br

***Abstract*—The publish-subscribe communication paradigm is widely used in systems that require a loosely coupled asynchronous form of interaction. The PubSubHubbub protocol is a publish-subscribe protocol for the Web that involves publishers, subscribers, and hubs, which are the intermediate elements between publishers and subscribers. However, in the original implementation of the protocol, unnecessary computation and network traffic occur as the sequence of exchanged messages to subscribers to retrieve a message is not optimized. In this paper, we present a lightweight version of such a protocol, named Light-PubSubHubbub, by introducing the following changes to the communication process: (i) the publisher no longer needs to publish updated messages in a Web topic and then notify the hub since the messages are published in the hub itself; (ii) it uses the REST architectural style in order not to couple publishers, subscribers, and the hub; (iii) XML is the default format of the messages. This paper also presents the results of experiments comparing Light-PubSubHubbub with the original PubSubHubbub protocol and the JMS technology for asynchronous messaging. The obtained results have shown that Light-PubSubHubbub takes less time to answer to the client than PubSubHubbub and JMS.**

Keywords-asynchronous communication; publish-subscribe; PubSubHubbub; Light-PubSubHubbub

I. Introduction

In the traditional way of the client-server communication, the server is the main element involved in the communication that receives and handles requests from clients. Nevertheless, such a model has shown to be significantly inefficient in situations in which data is frequently updated or such frequency is undetermined. In this case, the client needs to periodically send requests (synchronous) to the server to obtain the data and to check for updates. Such method is called *polling* [1], and although it meets the purpose of obtaining updates, it is not appropriate for situations when data are updated with an unknown frequency. Fig. 1 illustrates an example in which a client interested in updated data periodically makes requests to check for updates on the server. However, in this example, the client only gets an update on the third request; so, the first, second, and the fourth requests would be unnecessary because they do not provide any new information.

Although such a method makes the communication process between the client and the server simpler, it raises the question about the ideal period of time for making such requests. If a very large time period is chosen, the time for obtaining an update may be high and then it is possible to use

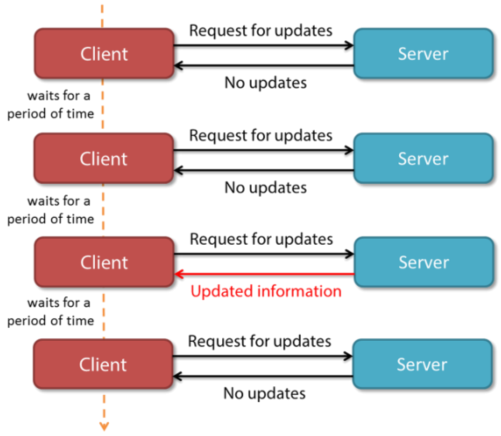


Figure 1. The polling method for obtaining information.

outdated information while an updated one is available. On the other hand, if the requests are performed in a very short period of time, unnecessary network traffic may be generated since the information may not be updated in such short time period [2]. In order to solve this problem, the *PubSubHubbub* protocol [3] was developed for dealing with event-oriented asynchronous requests [4] based on the *Publish-Subscribe* client-server communication model, in which *publishers* are responsible for sending messages to be consumed by *subscribers*. PubSubHubbub introduces a new element in such communication model called *hub*, which works as an intermediary between the publisher and subscriber elements. However, in the original implementation of such a protocol, the publisher needs to notify the hub that it has published an updated message in a Web *topic*, so that an additional processing must be performed by the hub in order to retrieve this new message and then forward it to the subscribers, thus generating unnecessary computation and network traffic. These limitations have motivated us to perform adaptations in the PubSubHubbub protocol to reduce its complexity, thus resulting in a lightweight protocol called *Light-PubSubHubbub*. In our approach, the publisher no longer needs to notify the hub that it has published a new message since publishers directly send the updated message to the hub instead of the Web topic. Therefore, no additional actions are performed to retrieve the published messages.

This paper is structured as follows. Section 2 gives an overview of the Publish-Subscribe communication model. Section 3 introduces the PubSubHubbub protocol. Section 4 presents the Light-PubSubHubbub protocol, resulted from adaptations in the original PubSubHubbub protocol. Section

5 presents a preliminary evaluation of the proposed protocol. Finally, Section 6 contains final remarks and future works.

II. The Publish-Subscribe Model

An *event* can be defined as a change in a state [4]. For example, when a person enters in his/her house, this action means a state change, i.e., an event. Events can be detected and dealt with applications through an *event-driven architecture* (EDA). Technically, such an approach enables the development of applications in which events trigger messages to be sent to independent modules of the application in an asynchronous way and according to the occurrence of such events.

In this context, the Publish-Subscribe model was developed as a model to deal with asynchronous messages in which *publishers* are responsible for sending messages that are consumed by *subscribers*. A great advantage of such model is the decoupling among its elements since publishers do not have knowledge about the subscribers registered for receiving their messages. However, the subscribers are able to choose what messages they want to receive from the publishers. Furthermore, subscribers receive only a subset of all published messages. The process of selecting such messages to be sent to the subscribers is called *filtering*, which can be *topic-based* or *content-based*. In a topic-based system, the messages are published in *topics*, which work as repositories of information of interest, so that subscribers will receive all messages published in the topic in which they have subscribed. In content-based systems, subscribers define constraints about the messages to be received, so that the messages are forwarded to them only if the message attributes or the content itself match the defined constraints.

In several systems that adopt the Publish-Subscribe model, there is an intermediary element called *broker* (or *event-bus*), which basically stores and forwards messages [5, 6]. In this kind of implementation, publishers publish messages in the broker, which forwards them to the subscribers that have been registered in the broker. There are also systems that do not use such intermediary element, so that publisher and subscriber share information (metadata) about themselves, thus forwarding messages based on the discovery of each other [6].

III. The PubSubHubbub Protocol

The *PubSubHubbub* protocol [3] is based on the Publish-Subscribe communication model and uses a broker element called *hub*. The hub is responsible for intermediating requests both from publishers (interested in distributing an updated information) and subscribers (interested in receiving the updates provided by the publishers), so that it receives update notifications from the publishers through an HTTP POST message, which informs the topic that has been updated. In a sequence, the hub makes a request to such topic in order to get the updated information. This request to the topic is performed through an HTTP GET message for obtaining updates, so that the updated information is forwarded to the subscribers through an HTTP POST message. Therefore, the PubSubHubbub protocol avoids that clients constantly perform checks for updates and it also eliminates the direct communication between the client and the server, which now is always intermediated by the hub (i.e., client–hub–server).

The PubSubHubbub protocol has four main elements:

1) The *topic* is the element in which the update information is published in the format of a feed by using the Atom [7] or Really Simple Syndication (RSS) [8] technologies. In general, the topic is publically available on the Web and can be accessed through an URL.

2) The *hub* is the element that works as an intermediary between the publisher and subscriber elements by: (i) receiving update notifications; (ii) accessing the topic provider in order to obtain updates; (iii) registering the subscribers, and; (iv) forwarding the updates to the subscribers.

3) The *publisher* is the element that publishes in the topic and is responsible for notifying the hub about the occurrence of an update. In the PubSubHubbub protocol, the publishers do not have to send the update to the hub. The publishers are only responsible for notifying it. The updates are published by the publisher as feeds, which is a data format used in communication transactions in which users frequently receive updated content.

4) The *subscriber* is the element that wants to receive updates regarding a given topic. In order to receive such updates, it is necessary that the subscriber has been subscribed in a topic of interest by making a request to the hub for subscribing to such topic. The hub will send to it the updates regarding the subscribed topic. The subscriber must be directly accessible through the network and identified by an URL.

PubSubHubbub works by performing three basic operations: (i) *discovery*; (ii) *subscription*, and; (iii) *publication*. In the discovery process the subscriber asks the publisher for a feed of a topic. Afterwards, the publisher sends the feed to the subscriber, which checks if there is an address regarding the hub used by the publisher for publishing updates in the topic and other important information, such as the update title and date when Atom feeds are used. If there is any reference to the hub in the feed sent by the publisher, then the subscriber can be subscribed to the referenced hub in order to obtain the updates whenever they are available. Otherwise, the subscriber must resort to the polling method or to other mechanism for obtaining updates regarding such topic since there is no reference to a hub in the feed, thus making impossible the use of the PubSubHubbub protocol.

In the subscription process, the subscriber requests the hub to subscribe to a topic by passing the address of the topic and the necessary information for sending the updates to the subscriber, and the hub confirms the subscription to the subscriber.

Fig. 2 illustrates the publication process in which the publisher publishes in the topic and immediately notifies the hub about the update by passing the address of the topic. In turn, the hub consults the address passed by the publisher and obtains the updated information for forwarding it to the interested subscribers. In such a communication model, the update of information requires the following actions illustrated in Fig. 2:

(1) the publisher publishes a new information in the topic;
(2) the hub is notified about the update in the topic;
(3) the hub requests the topic about the new available information;
(4) the hub receives the new information from the topic, and;
(5) the update is distributed to the interested subscribers.

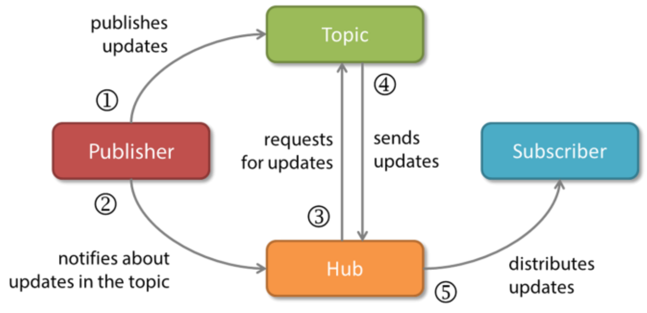


Figure 2. Processes performed by the PubSubHubbub protocol.

In addition, PubSubHubbub specifies operations based on the REpresentational State Transfer (REST) [9] architectural style, as means of establishing connections and requesting services, whether subscription or even publication requests. By using REST, PubSubHubbub can establish communications between hub, publishers, and subscribers by using only the HTTP protocol and several representation formats (e.g., XML, JSON [10] or plain text) without additional abstractions as in the SOAP protocol [11] for Web services.

IV. Light-PubSubHubBub: An Adaptation of the PubSubHubBub Protocol

In PubSubHubbub, the publisher is not responsible for sending information to the hub when it is published. It is only responsible for notifying it. Afterwards, the hub makes a request to the topic in order to obtain the updated information through the informed URL. In the adoption of this model, some shortcomings can be observed, such as the unnecessary computation performed by the hub, which must access the topic at each publication, and the generation of unnecessary network traffic because the updated information must go to the topic and then be retrieved by the hub, as well as the possibility of the topic being unavailable when it is accessed.

In this perspective, the original PubSubHubbub protocol was modified, resulting in the *Light-PubSubHubbub* protocol [12]. In this new proposal, the hub is not responsible for accessing the topic (on the Web) in order to obtain the updates, thus eliminating the need of publically accessing the topic through an URL. In the publication request to the hub (implemented by following the REST architectural style), publishers send information to the topic that must be previously registered in the hub, not in a server on the Internet, so that the hub must forward the updated information to the subscribers to such topic.

In the Light-PubSubHubBub proposed protocol, the publisher sends the updated information and the identifier of a topic that is registered in the hub. Next, the hub checks if the passed identifier of the topic is already registered and if there are subscribers interested in such publication. If true, the hub sends the updated information to the subscribers that are registered for receiving it. In this new perspective, the communication process is as follows: in the publication request to the hub, publishers directly send the updated information to the hub by following the REST architectural style; next, the hub forwards the updated information to the subscribers.

Another change that was performed over the original PubSubHubbub protocol refers to the used topic. PubSubHubbub extends the Atom and RSS protocols by using them as means of obtaining updates about information hosted in a server on the Web. Nevertheless, as the publisher is used for sending update data to the hub in the implementation of the Light-PubSubHubbub protocol, it was observed that the Atom and RSS technologies originally used for sending information to the hub could be easily replaced by XML-based messages (extensively used for message exchanges in the Web) since additional information (e.g., the update date in the Atom protocol) would not be necessary because the hub has now control over the topic. Therefore, the hub receives a publication and forwards it to the subscribers, thus bringing a greater flexibility to the Light-PubSubHubbub protocol in terms of the message format (that can be represented as XML, JSON, plain text, etc.) since there is no restriction regarding it. However, the subscriber must know the message format in order to suitably and correctly parse it.

It is important to highlight that the hub just works as a distributer, i.e., it does not need to know the object format since it only receives and forwards a string that must be parsed by the subscribers. Hence, if the messages are represented in the XML format, for example, the transformations to (*marshaling*) and from (*unmarshaling*) the XML format must be respectively performed by the publishers and subscribers. In turn, the hub can distribute information in any text-based format.

The following subsections detail the communication processes in the Light-PubSubHubbub protocol.

A. Registration

Before making any publication, the topic must already be registered in the hub. To do that, a publisher requests the registration of a new topic through an HTTP request to the RESTful service that is responsible for registering new topics. In the registration request, the hub checks if the passed identifier has not already been registered and then makes and confirms the registration.

The registration process regarding a new topic is performed through an HTTP PUT request to the URL regarding the RESTful service that registers new topics in the hub. The HTTP PUT request for registering a new topic in the hub is made by the publisher to the following URL:

```
http://<hub’s IP address>:<hub’s port>/Hub/register
```

The identifier of the topic must be in the body of the HTTP message. Fig. 3 shows an example of an HTTP message sent to the hub aiming at registering a topic with the *sports* identifier. In Fig. 3, lines 1 to 7 correspond to the header of the HTTP message and line 9 corresponds to the body of the message with the identifier of the topic.

```
1: PUT /hub/register HTTP/1.1
2: Content-Type: text/plain
3: User-Agent: Java/1.6.0_43
4: Host: 127.0.0.1:8084
5: Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
6: Connection: keep-alive
7: Content-Length: 6
8:
9: sports
```

Figure 3. Example of HTTP PUT message sent to the hub for registering a topic with the *sports* identifier.

B. *Subscription*

The subscriber must be registered to receive the updates regarding its topics of interest. In this perspective, the subscriber sends to the hub an HTTP request to the RESTful service responsible by such requests and passes as parameters: (i) the identifier of the topic of interest, which is requested by the hub in order to identify which updates will be sent to the subscriber, and; (ii) an address and a port used for identifying to where the updates will be sent. Therefore, each subscriber is uniquely identified by a triple composed of its IP address, identifier of the topic of interest, and the port in which it will wait for the notifications. After receiving a new request for subscription, the hub checks if the identifier of the topic is already registered; if true, the informed address and the port are registered as interested in receiving updates regarding such topic.

For a new subscription, it is made an HTTP PUT request to the URL regarding the subscription in the hub:

```
http://<hub's IP address>:<hub's port>/Hub/subscribe
```

The body of an HTTP PUT request for new subscriptions must contain the identifier of the topic of interest, and the address and the port for receiving updates. Fig. 4 illustrates an example of an HTTP PUT request sent to the hub in order to make a subscription to the topic with the *sports* identifier. In Fig. 4, lines 1 to 7 correspond to the header of the HTTP message and line 9 corresponds to the body of the message with the identifier of the topic of interest, and the address and the port for receiving updates.

```
1: PUT /hub/subscribe HTTP/1.1
2: Content-Type: text/plain
3: User-Agent: Java/1.6.0_43
4: Host: 127.0.0.1:8084
5: Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
6: Connection: keep-alive
7: Content-Length: 57
8:
9: {"address":"127.0.0.1","port":"50355","topic":"sports"}
```

Figure 4. Example of HTTP PUT message sent to the hub for subscribing to the topic with the *sports* identifier.

C. *Unsubscription*

If a subscriber does not want to receive anymore updates regarding a topic, it is necessary to make an HTTP request to the RESTful service responsible for cancelling such action. As a client may be registered for receiving updates regarding more than one topic, it is necessary to specify the information about the client and the identifier of the topic. In order to perform such operation, the subscriber sends the identifier of the topic, and its address and port, so that the hub removes such client from the list of interested subscribers.

In order to cancel a subscription, an HTTP DELETE request to the URL regarding the RESTful service responsible for such operation is made by passing through such URL the information that uniquely identify the resource to be deleted. Unlike the previous operations in which the parameters can be directly sent in the body of the HTTP message, the information for this operation is passed in the URL itself due to limitations of the HTTP DELETE request. Such URL is as follows:

```
http://<hub's IP address>:<hub's port>/Hub/
unsubscribe/?address=<subscriber's IP address>
&idTopic=<topic of interest>
&port=<subscriber's port>
```

Fig. 5 illustrates an example of an HTTP DELETE request to the hub aiming at unsubscribing a subscriber from the topic with the *sports* identifier. In Fig. 5, lines 1 to 6 correspond to the header of the HTTP message, which can be empty because the information needed to cancel the subscription have already been sent in the URL of the request.

```
1: DELETE /hub/unsubscribe/?address=127.0.0.1&idTopic=sports&
2:     port=14746 HTTP/1.1
3: User-Agent: Java/1.6.0_43
4: Host: 127.0.0.1:8084
5: Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
6: Connection: keep-alive
```

Figure 5. Example of HTTP DELETE message sent to the hub for unsubscribing to the topic with the *sports* identifier.

D. *Publication*

The publication process only happens when the topic that is being updated is already registered in the hub, otherwise a "topic not found" exception is thrown. In order to make a publication, a publisher sends to the hub the identifier of the topic and the value to be published. The hub checks if the passed identifier is already registered, and if true, it stores the information contained in the request.

The publication is performed through an HTTP POST request to the hub containing the identifier of the topic of interest that must be updated. The request URL is as follows:

```
http://<hub's IP address>:<hub's port>/Hub/publish/
<identifier of the topic>
```

After publishing, the hub sends the updated information to all subscribers registered for the current topic by using their respective address and port that were previously registered when subscribing. If there is no registered subscriber, the information is immediately discarded.

The body of the HTTP message for publishing a new content regarding a given topic must contain the value for update, which can be a string or even a XML representation of an object that must be parsed by the subscribers. Fig. 6 illustrates an example of an HTTP POST request to the hub in which the publisher wants to publish information in the topic with the *sports* identifier.

```
1:  POST /hub/publish/sports HTTP/1.1
2:  Content-Type: text/plain
3:  User-Agent: Java/1.6.0_43
4:  Host: 127.0.0.1:8084
5:  Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
6:  Connection: keep-alive
7:  Content-Length: 80
8:
9:  The 2014 FIFA World Cup will take place in Brazil from 12 June
10:     to 13 July 2014
```

Figure 6. Example of HTTP POST message sent to the hub for publishing information in the topic with the *sports* identifier.

In Fig. 6, lines 1 to 7 correspond to the header of the HTTP message, and lines 9 and 10 correspond to the body of the message that represents a new information to be forwarded to the subscribers. In such example, the update is regarding a message as a string.

V. EVALUATION

A. *QoMonitor*

The conducted case study consists of a ubiquitous oil and gas application that illustrates the need of monitoring the Quality of Service (QoS) and Quality of Context (QoC) of the services used by it. For monitoring such services, the *QoMonitor* [13, 14] system assesses, monitors, and makes available QoS and QoC metadata regarding services to be used by clients such as middleware platforms, Web services, applications, etc. *QoMonitor* handles synchronous and asynchronous requests from clients, both returning QoS and QoC metadata regarding a given service or a set of services. In synchronous requests, *QoMonitor* receives a request, processes the information, and answers to the client, i.e., the response time of such operation is the time for transporting the request/response over the network and the time for processing the request. In asynchronous requests, *QoMonitor* receives a subscription request, processes the information, and waits until a particular event (return condition) happens, and then asynchronously responds to the client, which needs to provide means of receiving responses from *QoMonitor*. To do that, the original implementation of *QoMonitor* uses a Java Message Service (JMS) [15] topic for forwarding the result of the subscription to the client when *QoMonitor* publishes in such topic. More details can be found at the URL http://consiste.dimap.ufrn.br/projects/lightpubsubhubbub/icsea2013.

However, the JMS technology generates a coupling between the clients and *QoMonitor* since JMS only works when the client is developed by using the Java programming language. In this context, the Light-PubSubHubbub protocol could have a key role since it enables the asynchronous communication between the clients and *QoMonitor* without generating a coupling because Light-PubSubHubbub was developed as a Web service.

B. *Experiments and results*

The performed experiments were aimed to address the overhead due to the use of the Light-PubSubHubbub protocol in comparison with the original PubSubHubbub protocol and the JMS technology, when *QoMonitor* notifies its clients about the event (return condition) regarding the asynchronous request. In the experiments, five different computers were connected to the same wired LAN network (in order to minimize the influence of the network) according to the experimental setup shown in Fig. 7. In order to calculate such overhead, a time Web service was developed for sharing the current time among the client, *QoMonitor*, and the used topic (JMS or hub). When *QoMonitor* publishes the notification in the topic, the time service is accessed for retrieving the current time and this time is stored. Afterwards, when the client receives the notification from JMS topic or the hub, it accesses the time service and the obtained time is subtracted from the time retrieved by *QoMonitor*, thus resulting in the time spent by the topic for answering to the client.

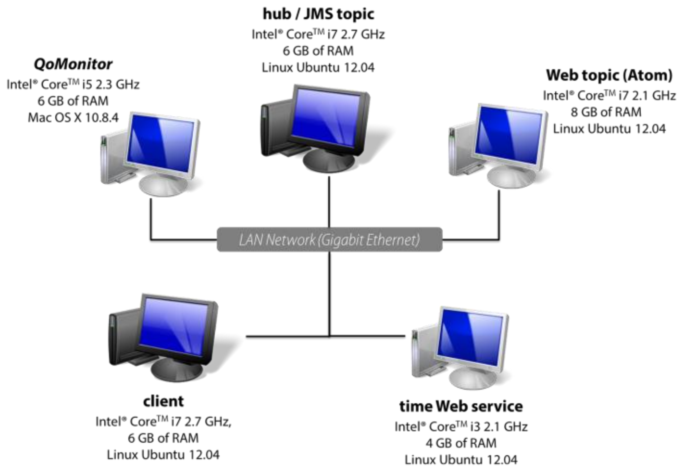


Figure 7. Infrastructure used in the evaluation of the Light-PubSubHubbub protocol compared with the original PubSubHubbub protocol and the JMS technology.

The experiments were conducted in three sequential phases. In the first one, the JMS technology was used by *QoMonitor* for communicating with the client, so that the client has performed an asynchronous request to *QoMonitor*, which has registered it in the JMS topic. Similarly, in the second and third phases, the PubSubHubbub and Light-PubSubHubbub protocols were respectively used, so that the client has performed an asynchronous request to *QoMonitor*, which has registered it in the hub. When the return condition was satisfied, *QoMonitor* answered to the client by using the used topic (JMS or hub). Twenty independent executions for the process of publishing and receiving the subsequent notification message were performed.

Table I presents the minimum, average, maximum, and standard deviation times spent (in milliseconds) by JMS, PubSubHubbub, and Light-PubSubHubbub within *QoMoni-*

tor. As can be observed in Table I, Light-PubSubHubbub takes less time to answer to the client than JMS and the original PubSubHubbub, thus resulting in a reduction of approximately 40% compared with JMS and 93% when compared to PubSubHubbub, on average.

TABLE I. TIME SPENT BY THE JMS TECHNOLOGY AND THE PUBSUBHUBBUB AND LIGHT -PUBSUBHUBBUB PROTOCOLS WITHIN QOMONITOR.

Technology	Minimum	Maximum	Average	Standard deviation
JMS	22.7671	30.6794	24.4792	1.7112
PubSubHubbub	173.4899	302.8242	209.085	33.2603
Light-PubSubHubbub	13.5101	19.3910	14.5962	1.3127

The considerable reduction observed when comparing Light-PubSubHubbub with the original version of the protocol is mainly due the fact that messages are directly sent to the hub instead of being posted to a Web topic, so that the hub can retrieve the message and then forward to the client, as in the original PubSubHubbub. Furthermore, as we have already argued, Light-PubSubHubbub does not generate a strong coupling between *QoMonitor* and the client since it was developed as a Web service, unlike the JMS technology that requires that the client be implemented by using the Java programming language.

VI. RELATED WORK

PubSubHubbub is a well-known protocol that has been used as a plug-in in several blog tools and content management systems (CMS) such as WordPress, Tumblr, Joomla, etc. Furthermore, there is also several works in the literature that have the same purposes of the PubSubHubbub protocol. For instance, the Java Message Service (JMS) [15] is a message-oriented middleware (MOM) that defines a set of interfaces that enable Java applications to communicate with each other. The JMS API enables asynchronism since it delivers the messages to consumers as soon as they are sent from the message producers, so that that consumers do not need to periodically request for the messages in order to receive them (as in the polling method). The JMS API also ensures that a message will be delivered one and only once, in a reliable way. The connection between consumers and producers can follow two basic models: (i) *point-to-point*, in which producers know consumers and directly deliver the message to them; or, (ii) *publish/subscribe*, in which publishers do not know subscribers and vice-versa since the communication among them is performed through the *JMS topic*, which receives the messages sent from publishers and forwards them to the interested subscribers. Since JMS is a Java technology, publishers and subscribers must be developed by using the Java programming language, thus generating a dependency in terms of technology, which does not happen in Light-PubSubHubbub.

Trifa [2] presents the Web Messaging System (WMS) protocol, which is based on the Publish-Subscribe model and is essentially similar to the PubSubHubbub protocol. WMS specifies the core functions of a Publisher-Subscribe system by using RESTful design patterns over HTTP interactions instead of developing a custom messaging protocol on the top of the HTTP protocol. In addition, it envisions a broker (very similar to the hub in the Light-PubSubHubbub protocol) that is responsible for storing the messages in an embedded database until their delivery to the subscribers. Despite of ensuring the delivery of the messages to the subscribers, this database storage may increase the latency for delivering the messages, as reported by the author.

In turn, Senn [16] uses the PubSubHubbub protocol in Wisspr (Web Infrastructure for Sensor Streams PRocessing), a Web-based framework for handling sensor data. Wisspr is built upon a Publish-Subscribe system in order to facilitate the development of event-driven and real-time processing applications for Web of Things by storing sensor data from different sources (e.g., mobile devices, home appliances, etc.) in a relational database. All sensor data are available from the PubSubHubbub protocol through a uniform RESTful interface, which enables to easily publish and consume data, as in Light-PubSubHubbub.

Another interesting publish-subscribe protocol is MobilePSM [17], which is intended to support mobile clients for publish-subscribe middleware. MobilePSM ensures that messages are not lost nor duplicated by temporarily storing them in a broker during the moving period, so that mobile clients can receive messages according to the sending order when a mobile client moves from one network to another or it is passively disconnected. This temporarily storage for providing reliability in terms of message delivering is an interesting feature that is not currently provided by Light-PubSubHubbub.

VII. CONCLUSION AND FUTURE WORK

In this paper, we presented Light-PubSubHubbub [12], a lightweight version of the PubSubHubbub protocol [3] to deal with asynchronous message exchanges in the Internet. This new version introduced the following changes: (i) the publisher no longer needs to publish updated messages in a Web topic and then notify the hub since the messages are published in the hub itself; (ii) it uses the REST architectural style in order not to couple publishers, subscribers, and the hub, and; (iii) XML is the default format of the messages. We performed experiments to compare Light-PubSubHubbub with the original PubSubHubbub protocol and the JMS technology [15] for asynchronous messaging, and the obtained results have shown that Light-PubSubHubbub takes less time to answer to the client than PubSubHubbub (a reduction of 93% on average) and JMS (a reduction of 40% on average). In addition, the implementation is not constrained to a specific technology since JMS uses the Java programming language, whilst Light-PubSubHubbub is implemented as a Web service.

As ongoing work, we are investigating how to address some current limitations of Light-PubSubHubbub. The first one regards to the temporary persistence of the messages when the subscriber is busy or down. Moreover, Light-PubSubHubbub does not provide any mechanism to ensure that only the owner of a topic can publish updated in such topic; in the PubSubHubbub protocol, publishers receive keys when registering for a topic and must use them in order to publish the updates. Finally, it is important to provide means of securing of the messages, in terms of cryptographing the exchanged messages.

ACKNOWLEDGMENT

This work is partially supported by the Brazilian National Agency of Petroleum, Natural Gas and Biofuels (ANP), through the PRH-22 Project.

REFERENCES

[1] H. Levi and M. Sidi, "Polling systems: Applications, modeling, and optimization", IEEE Transactions on Communications, vol. 38, no. 10, Aug. 1990, pp. 1750-1760.

[2] M. V. Trifa, Building blocks for a participatory Web of Things: Devices, infrastructures, and programming frameworks – PhD dissertation. Swiss Federal Institute of Technology Zurich, Switzerland, 2011.

[3] PubSubHubbub: http://pubsubhubbub.googlecode.com/ (access on September, 2013)

[4] K. Mani Chandy, "Event-driven applications: Cost, benefits and design approches", Gartner Application Integration and Web Services Summit, 2006.

[5] G. Cugola and H.A. Jac obsen, "Using Publish/Subscribe middleware for mobile systems", ACM SIGMOBILE Mobile Computing and Communications Review, vol. 6, no. 4, Oct. 2002, pp. 25-33.

[6] P.T. Eugster, P.A. Felber, R. Guerraoui, and A.M. Kermarrec, "The many faces of Publish/Subscribe", ACM Computing Surveys, vol. 35, no. 2, Jun. 2003, pp. 114-131.

[7] Atom Publishing Protocol: http://www.ietf.org/rfc/rfc5023.txt (access on September, 2013)

[8] RSS Specification: http://www.rssboard.org/rss-specification (access on September, 2013)

[9] L. Richardson and S. Ruby, RESTful Web services. USA: O'Reilly, 2007.

[10] JSON: http://www.json.org/ (access on September, 2013)

[11] Simple Object Acess Protocol (SOAP) 1.2: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/ (access on September, 2013)

[12] Light-PubSubHubbub: http://consiste.dimap.ufrn.br/projects/lightpubsubhubbub/ (access on September, 2013)

[13] C. Batista et al., "A metadata monitoring system for Ubiquitous Computing", Proc. of the 6th Int. Conf. on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM 2012). USA: IARIA, 2012, pp. 60-66.

[14] QoMonitor: http://consiste.dimap.ufrn.br/projects/qomonitor/ (access on September, 2013)

[15] Java Message Service (JMS): http://www.oracle.com/technetwork/java/jms/index.html (access on September, 2013)

[16] O. Senn, WISSPR: A Web-based infrastructure for sensor data streams sharing, processing and storage – Master's thesis. Swiss Federal Institute of Technology Zurich, Switzerland, 2010.

[17] T. Xue and T. Guan, "A protocol to support mobile computing for publish/subscribe middleware", Proc. of the 2012 Int. Conf. on Communication, Electronics and Automation Engineering, Advances in Intelligent Systems and Computing Series, vol. 181, G. Yang, Ed. Germany: Springer-Verlag Berlin/Heideberg, 2013, pp. 845-849.

Semantic Symbols Extraction Model for Emergency Hazard Map

Lijian Sun[1], Jie Zhao[2], Lihong Shi[1], Zheng Gong[2], Yi Zhu[1], Agen Qiu[1]

1. Chinese Academy of Surveying and Mapping
E-Government GIS Center
Beijing China
e-mail:sunlj@casm.ac.cn

2. National Administration for Code Allocation to Organization
Department of Data Processing
Beijing, China
e-mail:zjvsfd@sina.com

***Abstract*— Emergency event information released in map is necessary for emergency management and disaster reduction. A new method for emergency map symbols extraction from symbol collection based semantic analysis is presented in this paper. A novel map symbol semantic matrix is introduced to measure the degree of the semantic representation between the symbol meaning and the emergency event conception. The necessary content factors of emergency released in map are conduced by analyzing the emergency content construction and statistic result of the emergency content published by national disaster reduction center. The fuzzy comprehensive evaluation is proposed to extract the map symbol class. The sub function $f_i(k_i)$ is designed to confirm the risk of factor-ki . The simulation results verify the feasibility of emergency semantic model and its extraction model.**

Keywords-Symbol; Semantic Construction; Emergency; Fuzzy comprehensive evaluation.

I. INTRODUCTION

Natural disasters or emergency hazards occur frequently in the world [1], such as earthquake, typhoon, mud-rock flow, land-slip of the mountains, falling and the downthrow of the earth, etc. These are significant and realistic threats to people's wealth and life. Most of the natural disasters and emergency hazards are intimately related with spatial. To marked those hazard information including characters, impact range and the response processing by symbol on a map, it is of important practical and immediate significance to the disaster prevention and reduction. [2]. At present, researchers have done little work on this area [3]. The dissemination of emergency information processing system is mostly manual intervention, in a low automation level [2], so it is necessary to do some work in this area. The sematic of cartographic symbols includes spatical character and reference characters[4].

In this paper, a cartographic symbol extraction model on emergency hazard map n based semantic analysis is presented. A novel map symbol semantic matrix is introduced to measure the degree of the semantic representation between the symbol meaning and the emergency event conception. Further the fuzzy comprehensive evaluation and sub function are proposed to extract emergency map symbol from symbol library, by evaluating the relationship between the symbol semantic representation and the emergency event construction.

II. SEMANTICS RELATED TO EMERGENCY SYMBOL

Map symbols, composed of some graphics with different shape, size and color, are atlas language and could express geo-information effectively. Some researchers have done some work on map symbols linguistics, while most of them focused on the organization mode of graphic elements and symbol design. In this session, research on semantic representation of map symbol will be carried out by analyzing relationship between the entire semantic representation of the map symbol and the emergency content construction.

The map symbol for emergency release is thematic symbol. Spatial and representation are the basic attributes of those symbols.

The spatial character is the position on the map. According to the distribution mode of the object, the pattern of the map symbol spatial distribution includes dot, line, and poly. The dot distribution refers to the emergency whose spatial distribution mode is a point, for example, "January 2, 2008, 20 pm, a great fire occurred in U-City, D-Square", the D-Square is just a point in small scale map. The line distribution refers to the emergency whose spatial distribution mode is the line, for example, "January 6, 2008, Ice-run appeared in the Yellow River in Ningxia section, the whole length is about 234 Km". The poly distribution refers to the emergency whose spatial distribution mode is poly, for example, "from January 12,2008, a great snow storm fell in Anqing, Chizhou, Tongling district". The Anqing, Chizhou, and Tongling districts, which suffered from a great snow storm, are an area on a large scale map.

Symbol representation is to assess the relationship between the symbol and the emergency event, which is the ideographic expression. In this paper, the symbol representation measure is studied, not the symbol design. The symbol representation includes two aspects: symbol credibility and symbol (press) degree.

Definition: Symbol credibility: The real event is material, while the symbol is abstract, so there is gap between them. Symbol credibility (φ_{ij}) is used to measure this relationship.

φ_{ij} is the credibility for symbol i representing the case j. φ_{ij} is continuous not binary, $0 \le \varphi_{ij} \le 1$, i$\in$ symbol, j$\in$ case, φ_{ij} is dimensionless. The greater φ_{ij} is, the more little the gap is, that is the symbol is more similar to the event.

φ_{ij} is often set by expert's subjective judgment, it is a definition value.

Definition: Symbol (press) degree: Generally speaking, a different emergency is often represented with different degree's symbols, according to the emergency press. The class of emergency symbol could be set by different shape and graphics construction and the press degree of emergency map symbol could be set by different color. For instance, the level of different meteorological disaster symbols is distinguished by four color-blue, yellow, orange, and red (according to the regulation of meteorological disaster symbols for prediction, published by China Meteorological Administration in 2004) [6] .

TABLE I. METEROLOGICAL DISAER SYMBOLS COLLECTION (PART)

Meteorological disaster	**Risk Level**			
	Ⅳ **Slight**	Ⅲ **Serious**	Ⅱ **Magnitude**	Ⅰ **Destroy**
Typhoon	🟦	🟨	🟧	🟥
Rain Storm	🟦	🟨	🟧	🟥
Snow Storm	🟦	🟨	🟧	🟥
Cold Wave	🟦	🟨	🟧	🟥
Gale	🟦	🟨	🟧	🟥
Sand Storm	------	🟨	🟧	🟥
Drought	------	------	🟧	🟥
Frost	🟦	🟨	🟧	-----

In Table 1, the meteorological disaster level *Ⅳ*(Slight) is marked with blue, the level *Ⅲ*(Serious) is marked with yellow, the level *Ⅱ* (Magnitude) is marked with orange, and the level *Ⅰ* (destroy) is marked with red.

III. EMERGENCY SEMANTIC CONSTRUCTION MODEL

Emergency is an out-of-order incident, which people have little knowledge and information about [7]. A range of issues arising from the emergency are of ill-structured or unstructured problems [12]. If the emergency could not be responded effectively, it will lead to crisis. Emergency is often regarded as pre-crisis. Compared with general incidents, emergency incidents have the following three characteristics: emergent, severe ,and urgent.

Some linguists have done useful exploration and research on the analysis of the emergency content, such as Zeng Qingqing[5], who defined two information chains from the perspective of researchers—the main information chain and the secondary information chain. Yang Erhong analyzed the emergency content by setting the key words. The key elements in emergency content construction will be deduced by analyzing the relationship between the emergency meaning and the semantic representation of map symbols.

Event in context refers to language description for the special matter which people are concerned about, it belongs to the union meaning description [4]. Event is made up of event words and event parameters. In other words, the behavior generally described by verbs also includes event word, location, participants and so on.

Event word is to flag the property of the event, which is the key difference compared with other events. For example, "January 16, 2008 16:50 pm, one person was missing in a land-slip of the mountains, in Guxiang village, Xinshui town, Daning country". In this context, the event word is "land-slip of the mountains", which is the key word to distinguish the emergency. Generally speaking, a content collection may be confirmed by the type of the emergency when the event word appears.

Definition: Event word collect A:

$$A = \{a_1, a_2, , , a_n\} \tag{1}$$

where $a1, a2, ..., an$ are the elements of A, a_i is the event word. For example, the coalmine accident is defined as: Coalmine Accidents ={gas accident, collapse, colliery flooding, ...}.

The emergency hazard type should be certain from the context. If the type is not indexed in event word library, it could be pushed in the event word library, thus the event word library is open for extend.

Sometimes, the event word is not in context, for example, the context—"the houses have been razed to the ground, most of people was homeless", we knew the state of the house and the victims, but we could not detect the disaster type from the content, it may be earthquake, volcano or flood. This is called the event word missing, and the symbol could not be extracted from the emergency content when event word missing.

A. Event parameters

An integrated emergency should have a completed event word centric expression model that is given by:

$$a_i = f(p_{i1}, p_{i2}, , , p_{in}) \tag{2}$$

where p1, p2,,, pn are the event parameters to describe the emergency.

The components of the emergency event are always the mapping results from the part parameters in the whole event parameters. For example, the event parameters of the emergency- land-slip of the mountains includes :

" p_1 -Location",

" p_2 -Impact range",

" p_3 -The time of occurrence",

" p_4 -The duration",

" p_5 -The dead and hurt people",

" p_6 -The economic losing", etc.

For example, the context-"January 16, 2008 16:50 pm, one people was missing in a land-slip of the mountains, in Guxiang village, Xinshui town, Daning country", the event parameters in above include location, the time of occurrence and the number of missing.

B. *Event attributes*

Event attributes are used to describe the state of the event. The event modality, press degree, frequency and the state are focused on. The event modality is the possibility of emergency occurrence. The event modality mode is determined when the emergency has occurred, while for the predicted emergency, it should be marked by title or other form of annotation. For example, the context from the China Central Meteorological Station, "in the next 2-3 days, the 8th typhoon will generate in the northern of the South China Sea, the wind in typhoon center will be expected to reach 10-12 class. " is a predictive emergency. The event press degree is used to measure the hazardous extent of the emergency. It is in the content of the emergency sometimes, for example, "January 8, 2008, a serious traffic accident occurred in the Hexu expressway in Anhui province section". (The press degree in this emergency is serious). While sometimes it is not direct in the emergency content, for example," January 7, 2008, two were hurt in the traffic accident in 312 State Road in Yongshou country section". Sometimes the event press degree could be detected from the content if there is enough information in. The event frequency is the number of the emergency occurred. For example, "January, 20, 2008, two blasts occurred in a chemical factory, fortunately nobody was hurt", The event frequency of this blast is two times. The event state is used to describe the current state of the emergency incident, for example, "January, 10, 2008, a coalmine flooding accident occurred in K-Country, D-city. Now the rescue work is still on."

The statistic result of the emergency context for release with map symbols is as follows: (From national disaster reduction center, public)

TABLE II. THE SUMMARY STATICTIS RESULTS OF THE NUMBER OF EMERGENCY CONTEXT FOR RELEASE FROM JANUARY1,2008 TO JANUARY 27,2008,CHINA

Emergency	Event Word	Emergency Space		Emergency Attribute			
				Event extent		Time information	
		Location	Impact range	Numbered Infor	Extent Infor	Occurrence frequency	Duration Infor
Earthquake	11	11	2	11	0	11	0
Snowstorm	58	58	0	0	20	58	1
Fire	1	1	0	0	1	1	1
Over-water Accident	2	2	0	2	0	2	1
Traffic accident	4	4	0	4	1	4	0
Drought	4	4	0	4	0	4	0
Ice-run	1	1	1	1	0	1	0
Coalmine accident	1	1	0	1	0	1	0
Explosion	1	1	0	1	0	1	0
land-slip of mountains	1	1	0	1	0	1	0
Crash	1	1	0	1	0	1	0
Cold Wave	3	3	0	3	0	3	0
Total	92	92	3	29	22	92	3
Scale	100%	100%	3.26%	31.52%	23.91%	100%	3.26%

In Table 2, the event word, event space, event occurrence time are all emerging in the emergency context, combination of the research result of the map symbol, a conclusion is drawn that the emergency content for release in map should at least include the event word, space and occurrence time. The event word and the event space have to be marked on the map. The event occurrence time is sometimes marked out of the map, usually as title, annotation. The event extent is necessary for the level emergent.

IV. SOFTWARE MODEL DESIGN FOR SYMBOL EXTRACTION

The basic principle of the map symbols extraction algorithm is scattering the event according to the attributes of the elements. Then they can be converted to quantitative data according to the discrete elements and can be mapped to the symbol library through the mapping model . This paper will further extend this mapping as an evaluation.

The result of scattering event is the formation of an event object , the event object including (attribute 1 attribute 2, ... attribute n).Each symbol in the symbol library is associated with a semantic routing model .Semantic routing model is a table, which saves semantic categories and similarity of each symbol. The attribute of the event object can be into the evaluation index, the sign which gets a high score in this indicator is the symbol to be extracted. Then setting attributes (color, size, etc.) of the extracted symbol according to other semantic events in the object (degree of harm).The model is shown in Figure 2:

The Procedure Design Language (PDL) for map symbol extraction is as follows:

(1) Analyzing the emergency content, getting the elements collection $A = \{a_1, a_2,,, a_n\}$.

if (Event word $\notin A$ | Event space $\notin A$ | event occurrence time $\notin A$), A could not be released, then finish.

(2) Map symbol extraction algorithm:

if($\exists$ Event extent) $f1 = f$ (event word, event extent, symbol collection)

Else $f1 = f$ (event word, default event extent, symbol collection)

(3) if($f1 \neq \text{null}$), Symbol space position : $d = f($ Event space $)$, then $f1$ is marked at d .

(4) Mark the event occurrence time in map as label or title

or other illustration.

The algorithm and technical of position on map is quite mature, the event occurrence time is often marked as title or annotation. The map symbol extraction in step (2) is studied in this paper.

Define symbol as a bivariate function, that is,

$$S = f(B,V) \tag{3}$$

where B is the symbol ' s class and V is its extent.

Define Symbol semantic matrix, $B(i) = (\varphi_{ij})_{m\times n}$, φ_{ij} is the degree of credibility for symbol i substituting the case j .

Definition: Symbol attributes matrix, $L(i)_{1\times m} = (l_{kj})_{1\times m}$, that shows the relationship between attribute k and the class i. l_{kj} =1, when k consistent with j in class i, otherwise l_{kj} =0. For example, class i is {flood, snow storm, earthquake, volcano, explosion, ice-run,,,}, then the $L(earthquake)_{1\times m} = \{0,0,1,0,0,0,,\}$.

A. Symbol's class extraction

The fuzzy comprehensive evaluation, which is a method to classify the examples according by some indicators, is introduced to extract the map symbol class.

Suppose the count of event word is n, its attribute matrix is $L(i)$, (i=1,2,…n), then the extracted analysis matrix $X = |\eta_1, \eta_2, ..., \eta_r|^T$, $\eta_i = (L(i)A(i))^T$, where i=1,2,,,r. r is the count of symbol in symbol collection.

The processing steps are as follows:

1. Confirm the fuzzy relationship matrix Ri (including the membership function and the result), i=1,…,n.

2. Confirm the weighed distribution vector A, A= (a1,a2,,,,an).

3. Get the evaluation result Bi by blue processing, Bi = A ° Ri, Bi= (bi1,bi2,,,,bin).

where the samples belong to the class k*, when $b_{ik}^* = \max_k |b_{ik}|$.

B. Confirm the risk of symbol

The event degree is to explain the hazardous extent of the emergency, including the directly or indirectly information, indirect information, such as "7 people injured in a traffic accident, in Badong country, January 3, 2008", the directly information is that, "January 8, 2008, a serious traffic accident occurred in the Hexu expressway in Anhui province section".

(1) If the event press degree is not in the emergency context, define the event extent is the lowest.

(2) If the press degree l is in the context, the v is extracted directly. The higher the event extent level is, the more hazardous the emergency is.

(3) If the extent level is not direct in the context, while the number information about extent is in. Supposing that the information number k about extent is an indicators collection, that is, $K = \{k_1, k_2,,, k_n\}$ n is the count of the elements in k. Then, the sub-event extent to k_i could be confirmed according by the industry standard.

For example, the rainfall extent defined by China Meteorological Administrator is that: 1th grade is from 4.17mm/h to 8.33mm/h, 2 th is from 8.33mm/h to 16.67mm/h, 3 th is from 16.67mm/h to 33.33mm/h, 4 th is from 33.33mm/h to larger. The function is like that:

$$f_i(k_i) = \begin{cases} 1 & 4.17 < k_i \le 8.33 \\ 2 & 8.33 < k_i \le 16.67 \\ 3 & 16.67 < k_i \le 33.33 \\ 4 & k_i > 33.33 \end{cases} \tag{4}$$

The function (4) is shown in Figure 1.

The event extent with indicators collection k is that,

$$V = \max\{ f_1(k_1), f_2(k_2),,, f_n(k_n)\} \tag{5}$$

V. EXPERIMENTS AND ANALYSESE

Since the event word and press degree are necessary for emergency release. Suppose there are event words t1, t2, t3, t4 and press degree indicators k1, k2, k3, k4 in the emergency context. The simulation processing $f1 = f$ (event word, event extent, symbol collection) is as follows:

A. Symbol's class extraction

Six map symbols in collection as examples will be evaluated. After analysis from the symbol mapping table, the relationship result between the symbol semantic and the event word is as follows:

TABLE III. THE RELATIONSHIP BETWEEN SYMBOLS AND EVENT WORD

symbol \ event word	t1	t2	t3	t4
1	0.9	0.0	0.5	0.2
2	0.1	0.9	0.4	0.3
3	0.0	0.1	0.3	0.7
4	0.1	0.9	0.0	0.6
5	0.2	0.4	0.1	0.0
6	0.0	0.2	0.1	0.1

Then the matrix X :

$$x = \begin{bmatrix} 0.9, 0.0, 0.5, 0.2 \\ 0.1, 0.9, 0.4, 0.3 \\ 0.0, 0.1, 0.3, 0.7 \\ 0.1, 0.9, 0.0, 0.6 \\ 0.2, 0.4, 0.1, 0.0 \\ 0.0, 0.2, 0.1, 0.1 \end{bmatrix}$$

Suppose the evaluation factors collection is set as follows:

TABLE IV. THE EVALUATION FACTORS TABLE

level \ evaluation factor	t1	t2	t3	t4
I	0.3	0.3	0.3	0.3
II	0.6	0.6	0.6	0.6
III	0.9	0.9	0.9	0.9

Define that the bigger the value of the evaluation factor is, the better the expression effect is.

Confirm the membership and weigh:

Define a factor membership function for evaluation is like:

$$u_{ij} = \begin{cases} 1 & 0 \le d_i \le c_{ij} \\ \frac{c_{ij+1} - d_i}{c_{ij+1} - c_{ij}} & c_{ij} \le d_i \le c_{ij+1} \\ 0 & c_{ij} < d_i \end{cases} \quad (6)$$

where u_{ij} is the factor i that belongs to the membership degree j. d_i is the value of the factor i. c_{ij} is the criterion value of i with degree j, i = 1,2,3,4， j=1,2,3.

So, the symbol K 's membership matrix with all the evaluating factors is defined as $R^{(k)}{}_{4\times3} = (\mu_{ij})_{4\times3}$,k=1,2,…,6. The weight parameter matrix Ak = (a1,a2,a3,a4), where a_i is like:

$$a_i = \frac{x_i / a_i}{\sum_{i=1}^{n} x_i / a_i} \quad (7)$$

Fuzzy comprehensive evaluation is presented next.

To evaluate symbol K , the evaluation matrix is generated by synthesizing Ak and $R^{(k)}{}_{4\times3}$. In this paper, the average weigh model M(*,+) is adopted, that weighed all the factors. Then Ak is as follows:

$$B_{6\times3} = \begin{bmatrix} A_1 \circ R^{(1)}{}_{4\times3} \\ A_2 \circ R^{(2)}{}_{4\times3} \\ \cdots \\ A_6 \circ R^{(6)}{}_{4\times3} \end{bmatrix}$$, put the data into, then

$$B_{6\times3} = \begin{bmatrix} 0.23, 0.21, 0.56 \\ 0.39, 0.08, 0.53 \\ 0.36, 0.42, 0.22 \\ 0.06, 0.38, 0.56 \\ 0.81, 0.19, 0.00 \\ 1.00, 0.00, 0.00 \end{bmatrix}.$$

It shows that symbols 0 and 3 are better than others, that is B = 0 or B = 3.(0,3 are the row order) , B(3,2) > B(0,2), so B = 3.

B. *Confirm the risk of symbol*

From Eq.(4), the conclusion is drawn that,

$$V = \max\{ f_1(k_1), f_2(k_2), f_3(k_3), f_4(k_4)\},$$

where $f_i(k_i)$ is a linear classified function, usually defined by industrial standard.

After confirming B and V, according to (3), the target symbol could be calculated. Generally speaking, the function $f(B,V)$ is a mapping function to the symbol, B is the row, V is the col.

The electronic map with this method is shown in Figure 3 (dot symbol and line symbol in map).

VI. CONCLUSIONS

In this paper, the context and semantic construction about the emergency for release was studied. The map symbol semantic matrix is introduced to measure the relationship between the symbol semantic representation and the emergency event. According to the research, the necessary conditions for the emergency release in map are deduced. The fuzzy comprehensive evaluation is proposed to extract the symbol's class and the classified function for the factor of the event extent is used to confirm the event press degree. In this experiment, the symbol semantic matrix values are set by subjective definition, the value will be set by the standard or the expert database after the enough resource has been collected in future. By analyzing the example, the effectiveness and practice of the present experimental method are proved. The extension of the normalization of performance value and efficiency compared with other method would be our future direction.

ACKNOWLEDGMENT

This paper is partially supported by the China National Key Technology R&D Program with no. 2012BAH24B02.

REFERENCES

[1] N. Sun and L. Sh. Li, "The Study of Hierarchy Assessment Theory about Meteorological Disasters". 2007 Annual Conference of the Chinese Meteorological Society, July.2007, pp. 85-93.

[2] S. Stein, R. Geller, and M. Liu, "Why earthquake hazard maps often fail and what to do about it, "Tectonophysics, July 2012, pp. 1-48.

[3] X. Li and Z. Li, "Preliminary Discussion on the Theoretical Framework of the Earthquake Symbology System Building," Technology for Earthquake Disaster Prevention, July.2012, pp. 37-41.

[4] A. Schaff, "Introduction to Semantic," Pergamon Press, Oxford, New York, 1962

[5] Q. Zeng and E. Yang, "Event Anonotation and Analysis about the Content in Sudden Events Discourse," Advances of computational Linguistics in China, July 2009, pp. 600-605.

[6] China Meterological Administration, Symbols for meteorological disaster warning signal. China Meteorological Administration, 2004,(4),http://zwgk.cma.gov.cn/web/showsendinfo.jsp?id=1011.

[7] D.S. Mileti, "Natural Hazards and Disasters – Disasters by Design A Reassessment of Natural Hazards in the United States," Joseph Henry Press, Washington DC, 1999.

[8] M.Me, et al.," Geological disaster analysis and risk evaluation by GIS," Journal of Geosciences Translation, March.1995, pp. 72-79.

[9] A. Bagga, "Analyzing the Complexity of a Domain With Respect To An Information Extraction Task," Proceedings of the MUC-7, 1998,(6) http://www.muc.saic.com.

[10] National Disaster Reduction, Disaster Information. http://www.jianzai.gov.cn/.

[11] Z. Hu and H.W. Yan, Analysis on Linguistics Mechanism for cartographic symbols and its application. Geography and Geo-Information Science, January.2008, pp. 17-20.

[12] L.J. Sun, Y. Zhu, and X.Diaodong Liu, Research on the Theory of E-Government Emergency Information Publication Rapidly Based on Hierarchical Workflow. Geomatics World, June.2009, pp. 16-21.

[13] ACE.ACE Chinese Annotation Guidelines for Event. http://www.ldc.upemn.edu/Projects/ACE/docs/Chinese-Events-uidelines_ V5.5.1.pdf.2005C.

[14] X.H. Tan, The Application of Fuzzy Comprehensive Evaluation and Grey Assembly Classifying in lithologic Classification . Anhui geologic, April.1996, pp.71-76.

[15] L.N. Dang, F. Wu and Xuedong Li, Representation pf map symbols based on description method. Journal of Geomatics Oct, July 2007, pp.16-18.

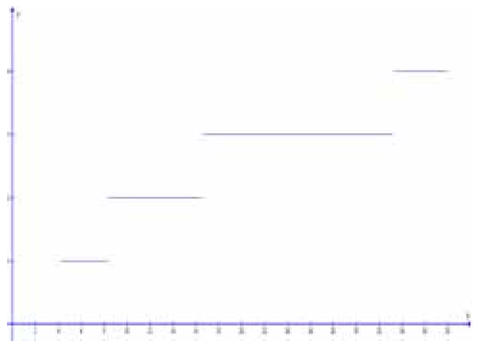


Figure 1. Example of Graphic for classified function

Figure 3. Emergency released in map by symbol

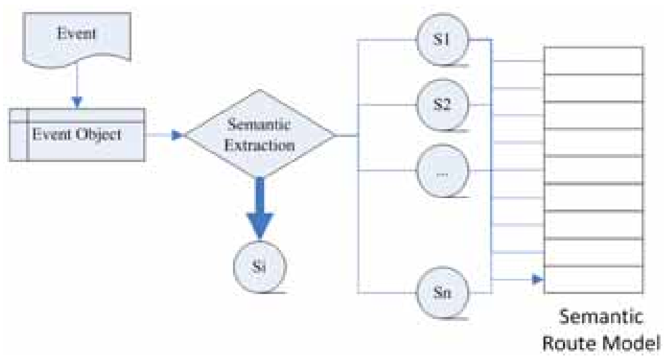


Figure 2. Software design model for symbols extraction

Interactive Hyperbolic Tree for Industrial size Software Product line Architecture

Abeer Khalid, Salma Imtiaz
Department of Software Engineering
International Islamic University
Islamabad, Pakistan
abeer.msse234@iiu.edu.pk, salma.imtiaz@iiu.edu.pk

***Abstract*—This Software Product line is an eminent part of software re-engineering field. Facilitation of software product line architecture with a more convenient method of representation mechanism results in efficiency with respect to time, cost, energy, etc. For this to be true, there is a need for information visualization techniques that represent true characteristics of software product line. This paper presents a study of information visualization technique which makes perception of data easy for interacting with the software product line architecture.**

Keywords-software product line architecture; information visualization; visualization representation

I. INTRODUCTION

Software product lines are known as a family of software systems, based on common and varying aspects of software products with immense complexity rooted in them. The present studies have suggested that architecture is the best suitable form there representation [24] [25]. Literature shows that representation mechanism, such as Unified Modeling Language (UML), matrix tables, conventional trees have so far been used in illustration of software product line architecture. But foremost, they have not depicted the characteristics of a software product line, which consequences in, not well attained results. For this problem to be tackled, an information visualization technique is the best suitable option [26].

In recent years, information visualization has taken grip of software engineering field by its sheer capability to enhance cognitive abilities for perceiving complex data [23]. Thought information visualization is a relatively new concept in the branch of software product line engineering, it can still be of immense help if a suitable visual structure plus its interactive visualization techniques are provided, as well said by Tufte “There are right ways and wrong ways to show data; there are displays that reveal the truth and displays that do not” [22].

A lot of work has been done in representation of software product line architecture data, with each technique having its pros and cons. The techniques presented so far are not scalable, traceable and they are not supporting evolution [10]. Present representation mechanisms for management of software product line architecture are not capable of handling the software product line architecture attributes and do not expose good visual structure attributes [26]. And thus, a visual structure technique is proposed, which is capable of conquering the attributes of a software product line architecture data, also that visual structure can be interacted upon; without being a static structure.

Hyperbolic trees are the visual structure devising the central piece for our Information visualization techniques. The criterion, on the bases of which hyperbolic tree structure was concluded as best fit structure, was obtained from attributes of software product line architecture and visual structure [26]. The criteria were set as “abstraction, hierarchy, traceability, scalability, evolution, visual content, and perception” [26]. Also, hyperbolic trees are chosen, for the fact that they “support exponential growth in the number of components with increasing radius” [5]. Hyperbolic tree stands on the basis that it has its root in the middle while its linked nodes and their children are spread apart. In short, this hierarchy depicts many generations of parents, their children, their siblings, in the same window snapshot without losing focus of the context [6]. The main feature of hyperbolic trees is their ability to be manipulated, without any regard to its extremely large hierarchy, which is much larger than conventional hierarchal structure. They have the ability to show 10 times as many nodes compared to other visual structures, and hyperbolic tree structure being more effective in providing navigation, without deviating from the context [5]. This takes care of our software product line architecture scalability issue to some extent.

This paper is organized in four sections: Section II is concerned with the problem and related work. Section III describes the visualization of the chosen visual structure. Section IV states the conclusion and direction for future work.

II. PROBLEM AND RELATED WORK

So far, representation of software product line architecture has used many techniques and notations (e.g., Matrix table conventional tree, then notations like UML, etc.). But, noticeably all these techniques are lacking in one way or another.

Literature suggests that a number of illustration mechanisms are used for representation of software product line data. Unified modeling language (UML) notations are a well-known representation form, and can be understood easily, with platform independence provided in them[16-21]. UML notations incorporated with natural languages are also used for representation of software product line data. Use case map path notations (UCM) are also used for representation of software product line data. The point to be notated is that all of the notations are good in some context [26], but they are not favorable for representation of software product line architecture data as a whole, where traceability links need to be visualized across the architecture as a whole, beside other factors.

Textual presentation is another representation form, which is used for SPL data [13] [14] [15]. But again, it is not feasible for the fact that, it is not scalable, no traceability links are present or visualized, keeping in mind that if no traceability, then evolution cannot be optimally utilized.

Matrix form is another type of notation which is used, as the literature suggests, for representation of software product line architecture data [9] [11] [12]. They are a good form of representation, but the problem with them is that they are not scalable for software product line architecture data; also, as with the above type of notation, traceability links are not visible.

Conventional trees are another type of representation form, whether they are vertical or horizontal tree [7], [9], [10]. They are the best form of presenting software product line architecture data. Here, the traceability links can be visualized for the whole context. However, they are not feasible because they are not a scalable structure, and also, when focusing on one aspect of the tree, the other parts of the hierarchy are obscured.

Cone tree is another form of hierarchal structure, which in 3D format is quite good; they overcome the prominent issues of the software product line architecture, namely scalability, plus visualization of traceability links [8], [9]. But, the problem of data obscuring is still present, meaning when focusing on one aspect of hierarchy, one does not see the full context in a single snapshot.

Tree maps are another form of hierarchal structure, which optimally utilize the screen space [7]. But the problem with this type of technique is that traceability links are not visible, also specifically one area of hierarchy cannot be focused on, without losing the grip on the context.

In sum, the shortfall of the above mentioned representation mechanism can be atoned by hyperbolic tree structure, based on the fact that its essence is favorable for software product line architecture data [26].

III. VISUALIZATION OF HYPERBOLIC TREE

The mapping of software product line architecture data on to hyperbolic tree is based on the fact that this visual structure is best suited for this job [10]. As defined in [5] and [6], hyperbolic trees support large hierarchies and their results have shown a preference towards the hyperbolic tree, as compared to conventional approaches. The authors of [5] and [6] also briefed about the implementation and the general features of their hyperbolic browser.

Here, their work has been translated for software product line architecture with enhancements included in it, based on the lack of presence of characteristics of software product line architecture. Also, the enhancements are derived from the perception capability of a human mind.

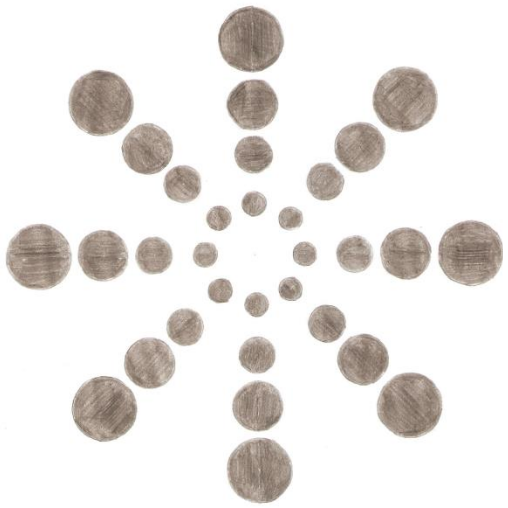


Figure 1. Based on Anstis (1974) work [3].

A. *Presenting "node"*

Each node is encompassed in a circle for displaying node information [5]. The circle does not interact with the circle of another node. The size of the circle would vary based on its generation level, e.g., if the node central to the core has size of 15cm, then, the next ring of nodes would have node with size of 10cm, which is 5cm short as compared to the parent and so on. The theory behind this logic is to show the distance factor giving the illusion of 3D depth factor. This is similar to the implementation in [3], where letter size is larger if the generation level is high. As shown in fig. 1, where outer most circle have large sized nodes, giving the perception that they are more close to the surface of the screen as compared to the other nodes; the illusion is that the size of the node decreases as they move further away from the surface of the screen. In fig. 2, Anstis [3] work has been translated onto the hyperbolic tree structure, where the inner most circles of nodes is giving the perception that they are closer to the surface of the screen. The next levels of generation of circle of nodes are positioned behind and so on.

When focusing on some point of a hierarchy then, the size of the nodes would vary, depending on the size of the parent node. The size of the parent node, and its child, and so on would become the same as compared to the other nodes at that specific time. Moreover, the positioning of the nodes with regards to the generation level would not be hindered when focusing on some part of the hierarchy.

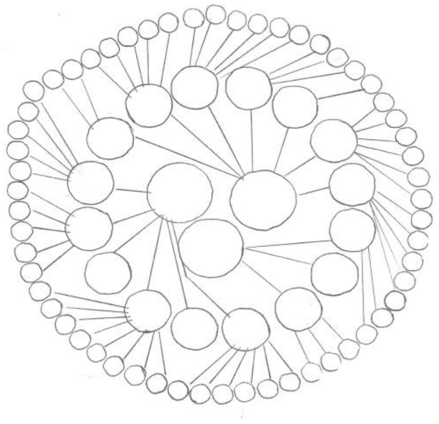


Figure 2. Hyperbolic tree structure with distance factor mapped on it.

B. *Generation level*

This feature has to be maintained for the sole reason that the perception of data for software product line architecture has a major hold. If the graph cannot maintain the level of placement of every node by their generation radius, then perceiving can be made quite difficult. The allotment of placement of nodes can be calculated by

$$\frac{\theta}{n} = x^{\circ} \Rightarrow x^{\circ} * s_i = y_j^{\circ} \,. \qquad (1)$$

where θ stands for total degree of angle, n implies total number of children, x° is the equal number of angle, s_i is number of children per node, and y_j° is the number of angle per node. Also, it should be stated that for each ring of nodes this equation is called for placement of next level of generation nodes. Then [5] presented in their article, equation for calculating the needed space from a parent to their child.

Lamping and Rao formula:

$$d = \sqrt{\left(\frac{(1-s^2)\sin(\alpha)}{2s}\right)^2 + 1} - \left(\frac{(1-s^2)\sin(\alpha)}{2s}\right) \qquad (2)$$

where α is angle between midline and edge of the subwedge and s is the desired distance between child and edge of its subwedge [5].Keeping in mind that even when focusing on some part of the hierarchy the level of generation gap should be maintained and not overlap at any point in time.

C. *Background landscape*

The background of it would be landscape, e.g., made up of peak mountain; the base of the mountain would be in green representing the grass, moving upwards it would merge with the color brown showing bare land, then moving upwards to color white representing snow. Figure 3 shows software product line architecture data translated onto the hyperbolic tree with human perception of real world environment kept in mind.

Perception of data is easy if the visualization is inspired from the real world environment and its objects known as “data landscape” in software terminology [4], based on the fact that skills used by human mind in interpreting the real world environment can be used in perceiving the visualization of “data landscape” [4].

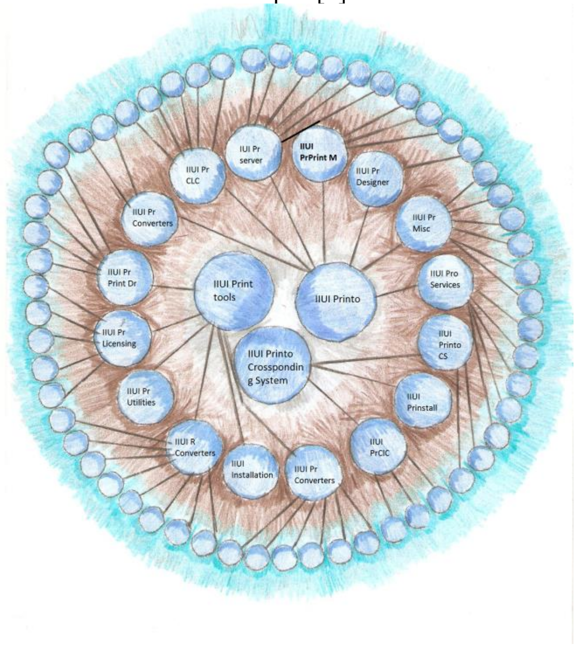


Figure 3. Perception of hyperbolic tree as real world object.

D. *Color aid*

The concept of “peak mountain” for the background, on which the hyperbolic tree would reside can be achieved with the help of color, as well said by Colin “that color helps in breaking camouflage” otherwise it would be very difficult to determine where or what a certain object is [4]. The use of color is not just about filling an image with color, but one has to bring it as close to real world objects as possible. In fig 3, the circles of nodes are filled with *Lambertian shading*, also the circles shown as objects, are C*asting shadow* on the mountain. Where *Lambertian shading* is known as a method for showing surface shape with the help of shading [4], meaning that if a mixture of color is not used then it is not possible to differentiate between the background and the overlaying objects on them. And C*asting shadows* theory is deduced from the fact that any real world subject can cast shadows either on itself or on the surface it is placed upon. This theory gives us the illusion of perception of height, of an object [4], stating that the specified object is at a height, above the ground that’s why it’s casting its own shadow on the ground; rather than being at the same level on the ground.

The nodes are also filled with the blue color and the text defining the node is in black color, which brings out the luminance contrast; which states that if the background is low saturation (light color), then the overlaid symbols must be of darker shade [4].

E. *“Affordance”device*

Taking Gibson’s *affordance* theory known as perceivable prospective for action [2] into consideration and translating it to our work, e.g., if the task is to bring second generation of children into focus, it would be highly recommended if "handles" are used [2]. As perceived by Houde, it is rather easy to perceive solution with the help of "handles" than arrows, etc. [1]. Here again, the focus is to bring forth human perception of real life objects, and use those skills as opposed to defining new ones.

IV. CONCLUSION AND FUTURE WORK

This paper starts with identifying the need for information visualization technique for the software product line architecture. It has mentioned the need for not just a good visual structure, but also the need for interaction with it. It further went on to explain the importance of hyperbolic tree and then presented enhancements to the concept of hyperbolic tree introduced by [5] and [6], for the sole purpose of establishing it as a fine means for the representation of software product line architecture data. Along the way, the perception of the human mind was kept in focus based on the rationale that nonfunctional requirement of software product line architecture can only be handled if perception of human mind is focused upon.

There is a need for testing this technique against previously used techniques for representation of software product line architecture. Our future work is based on this.

ACKNOWLEDGMENT

We would like to thank all our teachers and colleagues who helped. A.K, thanks MR. Mushtaq, Ms. Zafar, Mr. Iqbal, Mr. Hussian and Ms. Latif for their endearing support.

REFERENCES

[1] S. Houde. "Iterative design of interface for easy 3-D direct manipulation," Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM, Monterey, May, 1992, pp. 135-142. [Retrieved: October, 2013]. doi:10.1145/142750.142772

[2] J. J. Gibson. "The ecological approach to visual perception," Houghton Mifflin, Boston, 1979. (currently published by Lawrence Erlbaum, Hillsdale, NJ.)

[3] S. M. Anstis. "A chart demonstrating variation in acuity with retinal position," Vision Research, vol. 14, no. 7, July, 1974, pp. 589-592. [Retrieved: September, 2013]. doi:10.1016/0042-6989(74)90049-2

[4] C. Ware. "Information visualization:Perception for design," Morgan Kaufman Publishers, 2nd ed, 2004.

[5] J. Lamping and R. Rao. "Hyperbolic Browser:A focus+context Techniques for visualizing large hierarchies," Journal of visual languages and computing, vol. 7, no. 1, March, 1996, pp. 33-55. [Retrieved: September, 2013]. doi: 10.1006/jvlc.1996.0003

[6] J. Lamping, R. Rao, and P.Peter. "A focus+context technique based on hyperbolic geometry for visualizing large hierarchies," In Proceedings of the ACM SIGCHI Conference on Human Factore in Computing Systems(CHI '95), ACM, May, 1995, pp. 401-408. [Retrieved: October, 2013]. doi: 10.1145/223904.223956

[7] B. Johnson and B. Shnedierman. "Tree-maps: a space-filling approach to the visualization of hierarchical information structures," Visualization, 1991. Visualization '91, Proceedings, IEEE Conference on, vol., no., 22-25 October, 1991, pp. 284-291. [Retrieved: September, 2013]. doi: 10.1109/VISUAL.1991.175815

[8] G. G. Robertson, J. D. Mackinlay, and S. K. Card. "Cone trees: Animated 3d visualization of hierarchical information," Proceedings of the ACM SIGCHI Conference on Human factors in computing systems, ACM, 1991 pp. 189-194. [Retrieved: October, 2013]. doi:10.1145/108844.108883

[9] S. Card, J. Mackinlay, and B. Shneiderman. "Readings in Information Visualization - Using Vision to Think," Morgan Kaufmann, 1999.

[10] D. Nestor, L. O'Malley, A. Quigley, E. Sikora, and S. Thiel, "Visualisation of Variability in Software Product Line Engineering," in 1st International Workshop on Variability Modelling of Software Intensive Systems (VaMoS-2007), Limerick, Ireland, 2007. [Retrieved: October, 2013]. doi:10.1.1.136.9399.

[11] S. Ferber, J. Haag, and J. Savolainen. "Feature Interaction and Dependencies: Modeling Features for Reengineering a Legacy Product Line," Software Product Lines (SPLC2): Springer, vol. 2379, August, 2002, pp. 235-256. [Retrieved: September 2013]. doi: 10.1007/3-540-45652-X_15

[12] H. Ye, and H. Liu. "Approach to modelling feature variability and dependencies in software product lines," IEEE, vol. 152, June, 2005, pp. 101-109. [Retrieved: September, 2013]. doi: 10.1049/ip-sen:20045007

[13] S. G. Eick, J. L. Steffen, and E. E. Sumner. "Seesoft-ATool for Visualizing Line Oriented Software Statistics," IEEE Transactions on Software Engineering, vol. 18, no. 11, November, 1992, pp 957-968. [Retrieved: September, 2013]. doi:10.1109/32.177365

[14] A. van Deursen, M. de Jonge, and T. Kuipers. "Feature-Based Product Line Instantiation Using Source-Level Packages," Software Product Lines (SPLC2): Springer, vol. 2379, August, 2002, pp. 217-234. [Retrieved: October, 2013]. doi: 10.1007/3-540-45652-X_14

[15] K. C. Kang et al., "FORM: A feature-oriented reuse method with domain specific reference architectures," Annals of Software Engineering, vol. 5, no. 1, 1998, pp. 143-168. [Retrieved: September, 2013]. doi: 10.1023/A:1018980625587

[16] D. Muthig, and C. Atkinson. "Model-Driven Product Line Architecture," Software Product Lines (SPLC2): Springer, vol. 2379, USA, August, 2002, pp. 110-129. [Retrieved: October, 2013]. doi: 10.1007/3-540-45652-X_8

[17] D. Fey, R. Fajta, and A. Boros. "Feature Modeling: A Meta-Model to Enhance Usability and Usefulness," Software Product Lines (SPLC2): Springer, vol. 2379, USA, August, 2002, pp. 198-216. [Retrieved: October, 2013]. doi: 10.1007/3-540-45652-X_13

[18] S. Salicki, and N. Farcet. "Expression and Usage of the Variability in the Software Product Lines," Software Product-Family Eng (PFE-4): Springer, vol. 2290, Spain, October, 2002, pp. 304-318. [Retrieved: September, 2013]. doi: 10.1007/3-540-47833-7_27

[19] G. Halmans, and K. Pohl. "Communicating the variability of a software-product family to customers," Software and Systems Modeling, vol. 2, no. 1, March, 2003, pp. 15-36. [Retrieved: October, 2013]. doi: 10.1007/s10270-003-0019-9

[20] F. Bachmann et al., "A Meta-model for Representing Variability in Product Family Development," Software Product-Family Eng (PFE-5): Springer, vol. 3014, Italy, November, 2004, pp. 66-80. [Retrieved: October, 2013]. doi: 10.1007/978-3-540-24667-1_6

[21] D. L. Webber, and H. Gomaa. "Modeling variability in software product lines with the variation point model," Sci. Comput. Program., vol. 53, no. 3, December, 2004, pp. 305-331. [Retrieved: September, 2013]. doi: org/10.1016/j.scico.2003.04.004

[22] R. E. Tufte. “Visual explanation:Images and Quantities, Evidence and Narrative,” Cheshire, CT: Graphics Press, 1997.

[23] D. A. Norman. “Things that Make Us Smart,” Reading, MA: Addison-Wesley, 1993

[24] M. Sinnema, S. Deelstra, J. Nijhuis, and J. Bosch. "COVAMOF: A Framework for Modeling Variability in Software Product Families," Software Product Lines (SPLC3): Springer, vol. 3154, USA, August- September, 2004, pp. 197-213. [Retrieved: September, 2013]. doi: 10.1007/978-3-540-28630-1_12

[25] A. V. D. Hoek. “Design-time product line architecture for any-time variability,” Science of Computer Programming, vol. 53, no. 3, Neatherland, December, 2004, pp. 285–304. [Retrieved: October, 2013]. doi:10.1016/j.scico.2003.04.003

[26] K. Abeer, and I. Salma, “Evaluation of Visual structure for Industrial size Software Product Line Architecture,” Proc. of Eighth International Multi-Conference On Computing In The Global Information Technology, Think Mind, France(Nice), July. 2013, pp. 152-157. [Retrieved: October, 2013]. doi:iccgi_2013_7_40_10278

Creating a ITIL-based Software Incident Categorization Model for Measurement: A Case Study

Sanna Heikkinen, Antti Suhonen, Mika Kurenniemi, and Marko Jäntti
University of Eastern Finland
School of Computing
Email: firsname.lastname@uef.fi

***Abstract*—Many IT organizations have recognized incident categorization as a problematic subject because there are no general policies or guidelines for incident categorization. This leads to incident categorization usually being seen as an optional task for the specialists who handle incidents. This article presents the results of a case study that was carried out in an energy business unit of a Nordic IT service company. The research problem of this study is as follows: what type of software incident categorization model would be efficient and would also support ITIL-based continual service improvement? The results of this study consist of two parts: First, the software incident categorization (SIC) model which helps an IT organization to categorize incidents effectively and recognize the weak points of the software development process, and second, the provision of the lessons learned for improving incident categorization and measurement practices.**

Keywords*—*IT service management; ITIL; continual service improvement; incident management; software incident categorization model

I. Introduction

Managing incidents effectively is an essential operation for an IT organization and it usually affects several of the activities of the organization e.g., software development needs to change or fix an application or software in order to resolve an incident. IT organizations use different types of terms to define an incident (e.g., error, fix, bug, problem, programming error, user error, and hardware error), which may complicate understanding the meaning of the term, especially when the organization and its stakeholders are communicating about incidents. According to ITIL version 3 (Information Technology Infrastructure Library), an incident is an unplanned interruption to an IT service or reduction in the quality of an IT service [1]. In practice, an incident can be e.g., a software error, which prevents normal use of software, a malfunction in the printer, or a crashed database server. In this paper, the researchers use the description of ITIL v3 for the term ”incident”.

The ITIL is a set of good practices for directing and managing IT services and it can be tailored to any IT organization [2]. This study will focus on the Service Operation [1] and Continual Service Improvement (CSI) [3] lifecycle phases. One of the key processes of the Service Operation is incident management, which is responsible for managing the lifecycle of all incidents. According to the CSI ideology, an organization needs to measure the incident management process so that the organization can be sure that the process works effectively. The measurement data should be used to identify ideas for improvement to IT services or processes.

During the incident management process, incidents are arranged into categories. This is usually done by the service desk employees who are responsible for handling incident tickets through IT service management system. Incident categorization enables similar incidents to be tracked, which helps to recognize the weak points of services and processes. Although incident categorization is an important phase in incident management, there are no common incident categorization models, guides, or other best practices. This leads to the fact that organizations may create ineffective and unclear models for incident categorization and might mean that employees do not always understand the reasons and benefits which suggest why incident categorization should be performed in the first place. In practice, incident categorization should be user-friendly and explicit, and it should not slow down IT service management activities conducted by employees, such as diagnosing, escalating, and resolving incidents.

Incident categories are an important source of information when it comes to measuring and analyzing. The data that software incident categorization produces help IT organizations to identify the challenges and quality gaps in services and processes from the software lifecycle management point of view. Appropriate software incident categories allow the comparison of incident categorization data without country- or product- specific limitations. The organization’s future process improvement plans can also benefit from the data that software incident categorization produces. Ultimately, effective software incident categorization leads to increased customer satisfaction by improving product and service quality.

A. *Related Work*

Incident management is a central process for IT organizations and therefore many articles have been written about the subject from the software engineering and IT service management (ITSM) points of views. However, there have only been a few studies that have concentrated on incident categorization from the ITSM perspective. The present researchers exploited the following scientific articles while creating the software incident categorization model. In their paper Vipindeep and Pankaj [4] describe some of the common programming errors and poor programming practices that are often the cause of different types of bugs. Collofello and Balcom [5] introduce a causative software error classification scheme which emphasizes the entire software lifecycle and the causative attributes of the errors. In their paper Nakajo and Kume [6] researched the cause-and-effect relationship of software errors and human errors, which offers an appropriate framework for classifying the software errors. Lutz [7] used this framework when analyzing software requirement errors in safety-critical embedded systems. In their paper Leszak, Perry, and Stoll [8] describe a four-dimensional root cause classification. These four dimensions are human, review, project, and lifecycle. Owens, Womack, and Gonzalez [9] researched software error classification using a defect detection tool. Software errors were categorized into five classes: uninitialized memory read, array bounds write, array bounds read, free memory read, and free memory write errors. IEEE standard 1044-2009 [10] provides a uniform approach to classifying software anomalies, regardless of whether they occur within the project, product, or system lifecycle. Classification data can be used for a variety of purposes, including defect causal analysis, project management, and software process improvement.

B. Our Contribution

The main contributions of this paper are: 1) the software incident categorization (SIC) model which helps an IT organization to categorize incidents effectively and recognize the weak points of its software development process; 2) the provision of lessons learned for improving incident categorization and measurement practices.

The goal of this study was to design an appropriate and consistent incident categorization model which an IT organization could configure into its ITSM system. The purpose of the SIC model is to help IT organization to allocate incidents to a specific part of the software development process. In other words, the SIC model makes it easier to detect sections where customers have found incidents and which are not detected by the IT organization. The results of this study are mainly meant to be of benefit to the persons who are responsible for managing, measuring, and reporting IT services and IT service management processes (e.g., service owners, service managers, process owners, and process managers). This research does not address how the SIC model should be integrated into different ITSM systems. However, this integration should not be problematic with the systems that support ITIL v3 best practices because the SIC model was built on the basis of ITIL.

The rest of the paper is organized as follows. The research problem and methods are described in Section 2. The creation and validation of the software incident categorization model is covered by Section 3. The analysis of the findings, with lessons learned, is covered in Section 4. The conclusion in Section 5 summarizes the case study.

II. Research Methods

The research problem of this study is this: what type of software incident categorization model would be efficient and would also support ITIL-based continual service improvement? This study was a qualitative research study which was built using the case study research and action research methods. The research problem was divided into the following research questions:

- RQ1: What type of information can be used as a guide in creating an effective software incident categorization model?
- RQ2: How should the software incident categorization model be structured so that software-related incidents can be arranged effectively?
- RQ3: How should the software incident categorization model be validated?
- RQ4: How can incident categorization be used to support key CSI activities, such as measurement, reporting, and identifying the ideas for improvements?

During the case study, a researcher is an outsider, who observes and analyses an environment and makes notes by combining different data collection methods [11]. According to Baskerville [12], the action research method produces highly relevant research results because it is grounded in practical action, and it solves an immediate problem case while carefully informing theory. These selected methods support a situation where the researchers work together on a research project and their objective is to identify and solve problems in the IT organization's environment. The researchers used ITIL [2], and the ISO/IEC 20 000 standard [13] as theoretical frameworks in this study.

A. Case Organization and Data Collection Methods

The case subject of this study was an energy business unit which is part of a Nordic IT service company that provides solutions and services for Scandinavian energy companies. In 2012, the Nordic IT service company had around 17 000 employees operating in over 20 countries. The company's energy business unit is one of the research project's cooperation partners. This energy business unit will be referred to by the name Alpha for the rest of the paper.

The research was conducted in January 2013, using the KISMET (Keys to IT Service Management Excellence Technique) model as a roadmap to improve incident management practices. The KISMET model is presented in more detail in Suhonen's et al. research paper [14]. Multiple data collection methods proposed by Yin [11] were used during the study and the following data sources were used:

- **Documents**: meeting memos and process charts.
- **Archival records**: articles, incident categorization sets, and incident records.
- **Participatory observation**: meetings and discussions with managers (e.g., product, portfolio, development, release, and test managers).
- **Physical artifacts**: access to the intranet and to the IT service management system.
- **Semi-structured themed interviews**: interviews with five of the IT organization's staff members (senior software engineer, service desk specialists, and continuous service manager).

B. Data Analysis Method

This study was performed by using within-case analysis for a single organization. According to Eisenhardt [15], the within-case method typically involves detailed case study write-ups for each site and becoming familiar with the case as a stand-alone entity. The data analysis was performed collectively with the research group. The idea behind this collective analysis is to provide ”seeds for development” and to use their expertise in the analysis, as they know their specific fields best [16]. The triangulation used in this study allowed the researchers to be more confident about their results. Denzin [17] extended the idea of triangulation beyond its conventional association with research methods and designs. During the study the researchers used three forms of triangulation [17]: 1) data triangulation, which includes collecting data through several sampling strategies; 2) investigator triangulation, which refers to the use of more than one researcher in the field to gather and interpret data, and 3) methodological triangulation, which refers to the use of more than one method for gathering data. The research work was organized into chronological order by the phases of the KISMET model. The research work was validated during weekly meetings with Alpha’s representatives.

III. Results

In this section, the researchers will introduce the way in which the software incident categorization model was created in cooperation with the case organization and the research team. The research work consisted of five main phases: A) investigating the current state of incident management and planning improvement actions; B) designing a software incident categorization model based on ITSM practices; C) presenting the main categories and subcategories of the software incident categorization model; D) validating the SIC model, and E) presenting continual service improvement actions. These phases are described in the following subsections.

A. Investigating the current state of incident management and planning improvement actions

The kickoff meeting between the research team and the business unit Alpha was held in January 2013. At that meeting, the representatives of Alpha reported that they would like to improve and unify their unit’s internal measurement practices by designing a software incident categorization model.

The researchers analyzed the current state of Alpha’s incident management. During the analysis, the research team recognized a few challenges which implied to the team that appropriate improvement actions were needed. After that the researchers defined the improvement actions for Alpha and explained to them why executing these actions systematically is important (business benefit).

The recognized challenges: the researchers recognized that Alpha uses different incident categorization sets (sets of values for categorizing incidents). The lack of a consistent incident categorization set means that incidents are not categorized similarly inside Alpha. For this reason the same types of incidents may be arranged into different categories. This complicates the consistent measuring and reporting of different types of incidents. **Improvement actions:** Alpha requires an appropriate and consistent software incident categorization model in order to categorize incidents in a systematic way throughout the business unit. This model will help to analyze and compare different types of incidents and their frequencies inside Alpha (and between other business units if they implement the same software incident categorization model). **Business benefits:** by using an appropriate software incident categorization model, Alpha is able to design clear and measurable objectives for incident management. The measurement results can be used to identify areas or activities which cause delays in incident management. For instance, these results can show that the resolution times in network-related incidents are much longer than the resolution times for other types of incidents or lots of incidents were initiated during a testing phase (which may indicate that the testing is not executed properly). Regularly reviewing effectively categorized incidents on the basis of priorities and underlying causes could help to identify opportunities for continual service improvement, increase the quality of IT services, and improve customer satisfaction. A systematic model for managing improvement actions concerning IT services and IT service management processes have been presented in Heikkinen’s and Jäntti’s paper [18].

B. Designing a software incident categorization model based on ITSM practices

The researchers designed the software incident categorization model by using the ITIL technique [1], which can be applied to creating a complete set of incident categories. This technique contained the following steps:

1) Organize brainstorming sessions. Appropriate stakeholders should be invited to the sessions (e.g., service desk managers, incident managers, and problem managers).
2) Create the main categories for incidents by using the information collected during Step 1. Additionally, add an ”Other” incident category.
3) Test the main categories which were created in Step 2. Testing should last a sufficiently long period of time for the appropriate amount of data to be collected.
4) Analyze the data which were collected during the Step 3. The successfulness of the main category is determined by the number of incidents that have fallen into it. Additionally, analyze incidents which have been categorized as ”Other” incident. If the ”Other” incident category contains a large number of incidents, form new main categories for these incidents on the basis of similarities found.
5) Execute a breakdown analysis of the incident categories that have been created. The purpose of this analysis is to review the main categories and design appropriate subcategories for them.
6) Repeat Steps 2 to 5 for an appropriate period of time (approximately, from one to three months). Review the categories and subcategories regularly to ensure that they remain relevant.

The data sources that the researchers collected, analyzed, and used while executing these six steps are presented in Section II.

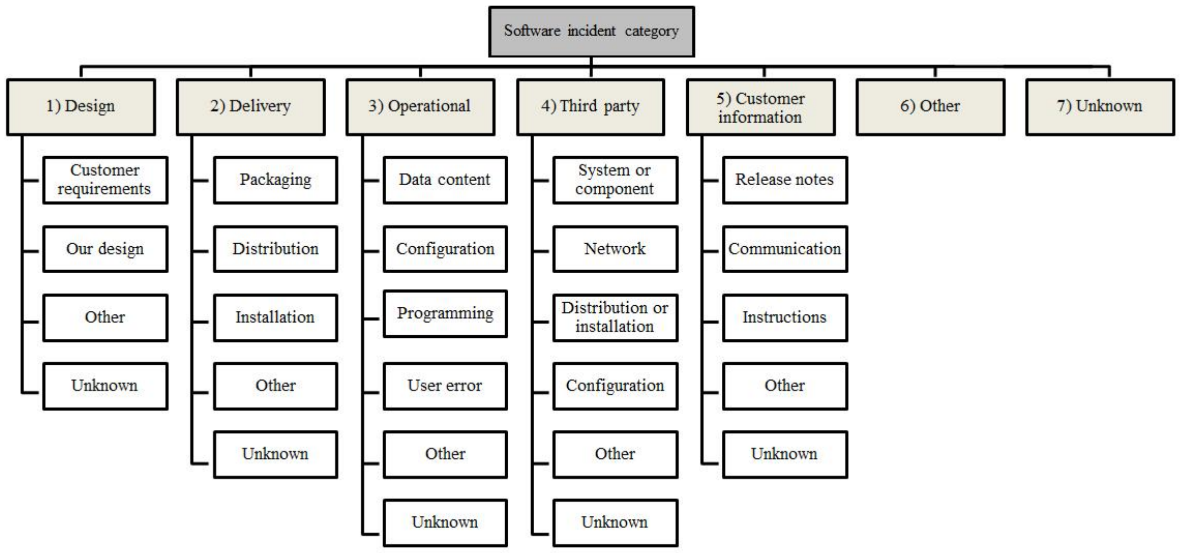


Fig. 1. The software incident categorization (SIC) model

C. Presenting the main categories and subcategories of the software incident categorization model

The software incident categorization model that was created offers a consistent and practical means of incident categorization. The SIC model is hierarchical and it consists of seven main categories and twenty-six subcategories. The model is not bound to any specific software or business unit. Figure 1 presents the structure of the software incident categorization model.

The model includes the main categories ”Other” and ”Unknown” (categories six and seven). Additionally, the main categories from one to five contain subcategories ”Other” and ”Unknown”. In practice, the ”Other” and ”Unknown” main categories and subcategories are meant to be used in the following way: the ”Other” category contains incidents that cannot be classified into the other categories and the ”Unknown” category will be used when the right classification category for the incident is not (yet) known. The list below presents the software incident categorization model’s main categories and subcategories in more detail:

1) **Design**: this main category contains incidents caused by customer requirements, improper translation of requirements into design, or the poor definition or inadequate specification of software.
   - **Customer requirements**: this subcategory covers incidents caused by inconsistent, incomplete, or incorrect customer requirements.
   - **Our design**: this subcategory covers software incidents caused by the improper translation of requirements into design. Incidents caused by the poor definition or inadequate specification of software also fall into this subcategory.
2) **Delivery**: this main category contains incidents that occur during software delivery or installation procedures.
   - **Packaging**: this subcategory covers incidents caused by software packaging.
   - **Distribution**: this subcategory covers incidents caused by software distribution.
   - **Installation**: this subcategory covers incidents caused by software installation.
3) **Operational**: this main category contains incidents that occur during the normal use of software (e.g., the software behaves incorrectly or it does not work with all inputs).
   - **Data content**: this subcategory covers incidents related to data management (e.g., database incidents, file handling incidents, and incidents related to measurement data).
   - **Configuration**: this subcategory covers incidents related to configuring the software.
   - **Programming**: this subcategory covers incidents related to programming errors. A programming error produces an incorrect or unexpected result, or causes software to behave in unintended ways (code may compile and run without error, but the outcome of an operation may produce an unexpected result).
   - **User error**: this subcategory covers incidents related to errors made by users. A user error results from a mistake made by a user.
4) **Third party**: this main category contains incidents that occur with the use of a third party’s software and hardware.
   - **System or component**: this subcategory covers incidents related to third party systems or components which do not behave as they were supposed to.
   - **Network**: this subcategory covers incidents related to the network.
   - **Distribution or installation**: this subcategory covers distribution and installation incidents caused by a third party.
   - **Configuration**: this subcategory covers incidents related to the configuring of the software caused by a third party.

5) **Customer information**: this main category contains incidents that are caused by incorrect or misleading information between the customer and the organization.
    - **Release notes**: this subcategory covers incidents related to release notes (e.g., customers feel that they have not been informed properly about the changes to hardware, software, or other components).
    - **Communication**: this subcategory covers incidents related to communication between a customer and the organization's employees (e.g., the service desk or support specialists).
    - **Instructions**: this subcategory covers incidents caused by written instructions, manuals, or training materials.
6) **Other**: this category contains incidents that cannot be categorized into the previous categories (categories 1 - 5). This category should exist because it helps to understand whether the SIC model works correctly and the categories that have been created are easy to use. This category also indicates whether other categories need expanding.
7) **Unknown**: this category will be used when the right category (1 - 6) for an incident is not yet known.

D. Validating the SIC model

The software incident categorization model that was created was validated by collecting data from Alpha's personnel (e.g., product area managers, problem managers, and service desk employees) using interviews and surveys. The following questions were used to validate the model:

- How does incident logging or managing appear to you in your job? Could you describe a typical incident situation?
- Are the software incident categorization model's main categories and subcategories appropriate and consistent, in your opinion?
- Is there a lack of any categories of the SIC model (e.g., are there any missing main categories or subcategories that you can think of)?
- In your opinion, is the software incident categorization model easy to use? Do you find it easy to discover the proper category for an incident?
- Are the descriptions of the main categories and subcategories appropriate and easy to understand?
- Have you found categorizing incidents challenging? If that is the case, please describe.
- What benefits can be achieved by using incident categorizing?
- Do you have any other ideas on how to improve the incident categorization?

The judge from the validations, Alpha's representatives were pleased with the model and its categories. The personnel were also keen to know when the model would be implemented and ready for use. The following comments were collected during validation meetings:

- The SIC model will help us see the most critical incident sources in software development. We will be able to identify the areas that cause most of the incidents and we can take appropriate countermeasures once these areas have been identified.
- Work was done earlier in small groups when our working practices were not a concern. Today, when work is done in cooperation with several groups, working practices need to be consistent if we want to measure and compare work e.g., from the quality point of view.
- Change and service request types of tickets need to have their own categorization models.
- Using the model (choosing the right main category and subcategory) may be challenging at first if appropriate documentation about the model is not available.
- The ”Other” and the ”Unknown” categories are useful in situations when it is hard to know the right subcategory for the incident, e.g., when an incident is sent to the service desk, which cannot know for sure what the exact incident subcategory is without the help of support specialists.
- What type of reports can be created by using the categorization data and how can these reports be exploited?

E. Presenting continual service improvement actions

Continual Service Improvement (CSI) aims to continually improve the effectiveness and efficiency of IT processes and services. Measuring the current performance of services and processes is an important factor when identifying improvement opportunities. The SIC model is closely linked to CSI by supporting the measurement of ITSM services and processes. With clear and measurable objectives (e.g., increase number of incidents related to software installation) organization is able to direct its ITSM improvement actions by using incident categorization data of the SIC model. The measurement data can be also used to identify flaws in e.g., incident, problem, and release management processes.

Before the implementation of the SIC model, Alpha should document and validate all the necessary instructions and training materials (e.g., example cases for every category). Alpha should also organize training for its employees to make sure that the SIC model is used properly. It would be wise to arrange regular checks on the SIC model after the implementation to ensure that the model works as expected. In practice, Alpha needs to review how well employees can use the categories and start appropriate improvement actions in case there arises any shortages during the SIC model implementation phase. All the identified opportunities for improvement should be logged in the CSI register, where they are evaluated, approved, prioritized, measured, and reported in a systematic way.

IV. Analysis

In this section, the researchers analyze the research findings in the form of the lessons learned. A source for each lesson is presented using the following abbreviations: DR = documents and archival records; PO = participatory observation, and PA = physical artifacts.

Lesson I: having and understanding consistent IT service management terminology is vital (DR, PO, PA). The researchers discovered that Alpha's personnel do not fully comprehend the actual meaning of an incident and how an incident differs from other support ticket types, e.g., service request. The issue was confirmed in January 2013, when the researchers noticed several dozen different definitions of incidents in the IT service management system. For this reason, Alpha has created several incident categorization sets. Using consistent ITSM terminology makes it easy to recognize what types of support tickets are incidents by nature.

Lesson II: there should be an appropriate and maintainable amount of incident categories (DR, PO). The incident categorization is more useful when it is kept simple. Adding new categories always has to be reasoned. This means that the categorization should help support groups to assign incidents to different categories. The categories should also support incident management analysis and reporting. Help desk personnel may find it difficult to decide which category is the right one if there are too many categories. Besides, if the number of categories grows too large, it is more likely that some of the categories would never be used. An unused category is useless and it has no value in reporting.

Lesson III: the category of the incident should be checked and updated if necessary during the lifecycle of the incident (DR, PO). The details available at the time of the incident categorization may be incomplete, misleading, or incorrect (the ”Other” and ”Unknown” categories in the SIC model are meant to be used in situations where the incident category is unclear). It is therefore important that the incident categorization is checked and updated if necessary, e.g., during the closure of the incident. The capability to track changes in incident category throughout the lifecycle of an incident may prove useful when looking for potential improvements (e.g., analyzing why the underlying cause of the incident was difficult to identify).

Lesson IV: automation is the key to logging incident information successfully (PO). The work of support group employees should not be slowed down by incident categorization. In practice, support group employees may need to complete several tasks to log an incident (e.g., fill mandatory input fields and choose the right values for drop-down lists). To save time and to make the incident logging process easier, employees may be unwilling to use the SIC model, which is why the incident logging process should be automated as much as possible so that employees' workload does not increase substantially. In addition, customer input for incident logging should be exploited whenever it is possible and convenient.

Lesson V: incident categorization supports continual service improvement (DR, PO). The organization should use reactive and proactive actions during the continual service improvement. From the reactive point of view, incident categorization makes it possible to recognize challenges and shortages in services. Proactively, acting in advance by executing appropriate procedures can be used to guide an organization in the desired direction. Managing and fixing recurring incidents is not effective. The organization should learn from previous incidents and take proper counter actions to ensure that the same incidents will not recur in the future. For example, incidents related to releases need to be monitored and analyzed for a sufficient period of time. The results and conclusions drawn from the analysis have to be recorded and reviewed to identify opportunities for improvement.

V. Conclusion and Future Work

The research subject of this study was an energy business unit, Alpha, which is part of a Nordic IT service company. The research problem of this study was this: what type of software incident categorization model would both be efficient and support ITIL-based continual service improvement? The research work consisted of five main phases: A) investigating the current state of incident management and planning improvement actions; B) designing a software incident categorization model based on ITSM practices; C) presenting the main categories and subcategories of the software incident categorization model; D) validating the SIC model, and E) presenting continual service improvement actions. The result of this study consisted of two parts: one, the software incident categorization (SIC) model which helps an IT organization to categorize incidents effectively and recognize weak points of the software development process, and two, the provision of the lessons learned for improving incident categorization and measurement practices.

The use of a case study and action research methods includes certain limitations. First, the research was performed with one organization, which means that the research work needs to be repeated in other organizations so that the results can be generalized. Second, the study was executed within a short period of time. A longer research period would have provided more detailed analysis of the SIC model and its work in practice. Third, the researchers could have conducted more validation meetings with Alpha's other business units to get a better understanding of whether the SIC model works as expected. Fourth, the purpose of this paper was not to research how the SIC model should be integrated into different ITSM systems. Since SIC model is built on the basis of ITIL v3 practices, it should be easily integrated to the systems which support ITIL. The management (e.g., adding, removing, and editing categories) of the SIC model should be also straightforward in organizations that are already familiar with ITIL best practices.

More studies are needed to investigate how the SIC model categories work and how the SIC model could be expanded to cover e.g., hardware-related incidents. Additionally, future research could concentrate on designing new models to support other ticket types (service requests and problems) by using the SIC model as a starting point.

Acknowledgments

This paper is based on research in the KISMET project, funded by the National Technology Agency, TEKES (no. 70035/10), the European Regional Development Fund (ERDF), and industrial partners.

References

[1] Cabinet Office, *ITIL Service Operation*. The Stationery Office (TSO), United Kingdom, 2011.

[2] OGC, *Introduction to ITIL*. The Stationery Office, London, 2007.

[3] Cabinet Office, *ITIL Continual Service Improvement*. The Stationery Office (TSO), United Kingdom, 2011.

[4] V. Vipindeep and P. Jalote, “List of common bugs and programming practices to avoid them,” 2005.

[5] J. S. Collofello and L. B. Balcom, “A proposed causative software error classification scheme,” 1985, pp. 537–546.

[6] T. Nakajo and H. Kume, “A case history analysis of software error cause-effect relationships,” 1991, pp. 830–838.

[7] R. R. Lutz, “Analyzing software requirements errors in safety-critical, embedded systems,” in *Proceedings of IEEE International Symposium on Requirements Engineering*, 1993, pp. 126–133.

[8] M. Leszak, D. E. Perry, and D. Stoll, “A case study in root cause defect analysis,” in *Proceedings of the 2000 International Conference on Software Engineering*, 2000, pp. 428–437.

[9] H. D. Owens, B. F. Womack, and M. J. Gonzalez, “Software error classification using purify,” in *Proceedings of International Conference on Software Maintenance*, 1996, pp. 104–113.

[10] IEEE Computer Society, “IEEE standard classification for software anomalies,” 2009.

[11] R. K. Yin, *Case Study Research: Design and Methods*. SAGE Publications ltd, 2003.

[12] R. L. Baskerville, “Investigating information systems with action research,” *Commun. AIS*, vol. 2, no. 3es, Nov. 1999.

[13] ISO / IEC, *ISO/IEC 20000-1:2011, IT Service management, Part 1: Service management system requirements*. ISO/IEC JTC 1/SC 7, 2011.

[14] A. Suhonen, S. Heikkinen, M. Kurenniemi, and M. Jäntti, “Implementation of the ITIL-based service level management process to improve an organizations efficiency: A case study,” in *The Eighth International Conference on Software Engineering Advances (ICSEA), Paper accepted*, 2013.

[15] K. M. Eisenhardt, “Building theories from case study research,” in *Academy of Management Review*, 1989, pp. 532–550.

[16] P. Eriksson and A. Kovalainen, *Qualitative Methods in Business Research*. SAGE Publications ltd, 2008.

[17] N. Denzin, *The Research Act in Sociology*, 1970.

[18] S. Heikkinen and M. Jäntti, “Establishing a continual service improvement model: A case study,” in *Proceedings of the 19th European Conference: Systems, Software and Service Process Improvement (EuroSPI)*, 2012, pp. 61 – 72.

Implementation of the ITIL-Based Service Level Management Process to Improve an Organization's Efficiency: A Case Study

Antti Suhonen, Sanna Heikkinen, Mika Kurenniemi, and Marko Jäntti
University of Eastern Finland
School of Computing
Email: firsname.lastname@uef.fi

***Abstract*—IT organizations' needs to reduce costs and maximize the efficiency and effectiveness of IT services have become essential factors for success. Processes, functions, and services require continual improvement in order to generate positive business results and high levels of customer satisfaction. This article presents the results of a process improvement case study carried out in the Information System Management (ISM) unit of the Finnish Tax Administration. The researchers focused on improving the ISM unit's service level management (SLM) process to increase employee and customer satisfaction. The research problem of this study is this: how to implement the Information Technology Infrastructure Library (ITIL) based SLM process to improve organization's efficiency? The main contributions of this paper are: 1) defining how to implement the ITIL-based SLM practices by using the Keys to IT Service Management Excellence Technique (KISMET) model to increase organization's efficiency, and 2) providing the lessons learned from improving SLM practices.**

Keywords*—*IT service management; ITIL; continual service improvement; service level management; service level agreement

I. Introduction

The Information Technology Infrastructure Library (ITIL) is a set of good practices for directing and managing IT services. The ITIL gives a detailed description of IT service management (ITSM) processes with comprehensive checklists, activities, roles, and responsibilities, which can be tailored to any IT organization [1]. ITIL version 3 approaches ITSM from the IT service lifecycle point of view. The IT service lifecycle consists of five phases: Service Strategy [2], Service Design [3], Service Transition [4], Service Operation [5], and Continual Service Improvement [1]. This study focuses on the Service Design and Continual Service Improvement (CSI) lifecycle phases, where the business perspective plays an important role. Continually improving services is vital for every IT organization because there is strong competition in business today and the IT services need to be continually aligned with the customer's needs. According to ITIL [1], CSI reviews, analyzes, and makes recommendations on improvement opportunities in each IT service lifecycle phase and ensures that these opportunities are identified and managed throughout the service lifecycle.

An ITIL-based service desk provides a single point of contact between IT organization and users on a day-to-day basis. This also means that a service desk is responsible for handling support tickets, which are managed via the IT service management (ITSM) system. The type of a support ticket can be an incident or a service request. An incident is an unplanned interruption to an IT service or reduction in the quality of an IT service [5]. In practice, an incident can be e.g., a software error, which prevents normal use of software, a malfunction in the printer, or a crashed database server. A service request is a formal request from a user for something to be provided (for example, a request for information or advice to reset a password, or to install a workstation for a new user) [5]. The service desk treats every support ticket as a separate entity (one logged support ticket to the ITSM system should cover one incident or service request). However, while handling same types of incidents, the service desk can create a link between these incident tickets by using an ITSM system. This practice enhances the efficiency of incident management.

Service level management (SLM) is a process of Service Design lifecycle phase in ITIL v3 [3]. The purpose of the SLM process is negotiating and documenting SLM agreements with appropriate stakeholders, and then monitoring and producing reports to follow these agreements [3]. According to ITIL, SLM agreements can be classified into three groups: service level agreements (SLA), operational level agreements (OLA), and underpinning contracts (UC) [3]. A SLA is made between an IT organization and a customer. An OLA is an agreement between two parts of the same organization and an UC is a contract between an IT organization and a third party.

Every SLM agreement contains rules. These rules define how an IT organization handles different types of support tickets. Usually a support ticket affects one or more configuration items (CI). A CI is any component or other service asset that needs to be managed in order to deliver an IT service. For example, a CI can be a service, hardware, software, a building, people or a formal documentation. Looking at SLM from ITSM system perspective, CIs are mandatory because they can be used to create links between SLM agreements, CIs and support tickets. For example, this practice makes it possible to create SLM agreement, which includes rules that only focus on incidents of the Service Alpha (an CI).

In practice, a rule in the SLM agreement is a combination of following attributes:

- **Configuration items (or an item):** which CIs (usually services) are affected by the rule?

- **Type of the support ticket:** which support tickets are affected by the rule?
- **Reaction and resolution times:** what kind of reaction and resolution times would be best for the selected CIs (these times might vary depending on the type of the support ticket)?
- **The priority of the support ticket:** how will the priority of the support ticket affect the calculation of reaction and resolution times?
- **The states of the support ticket:** which states will affect the calculation of reaction and resolution times?
- **Notification settings:** which people will be notified, when are the notifications sent, and what is the content of the notification?

In this context, the reaction time means the time in which work should start on a support ticket after the ITSM system has registered the ticket and the resolution time means the time in which a support ticket needs to be solved.

Well designed SLAs, OLAs, and UCs help to improve and maintain organization's efficiency. According to ITIL v3, efficiency is a measure of whether the right amount of resource has been used to deliver a process, service or activity. An efficient process achieves its objectives with the minimum amount of time, money, people or other resources [3]. In this paper, the researchers use the description of ITIL v3 for the term "efficiency".

The efficient handling of support tickets has become one of the essential tasks for IT organizations in the last few years. In practice, separate SLM agreements for different services enables that support tickets can be handled efficiently based on types and priorities of the tickets. This means that appropriate amount of resources (e.g., time, money, and right persons) can be allocated to handle tickets, which makes support ticket handling process efficient and has direct impact to organization's overall efficiency. It also seems that in the future there will be an increased number of support tickets, which customers will expect to be solved quickly and effectively. For these reasons many organizations have started defining and designing SLAs, OLAs, and UCs.

CSI has a strong interface with the SLM process. CSI reviews and analyzes SLA, OLA, and UC reports and if those reports indicate any deviations (e.g., the resolution times for support tickets being exceeded) CSI starts appropriate improvement actions. These actions should follow the ISO/IEC 20 000 standard [6]. ISO/IEC 20 000 is an international standard for ITSM. ISO/IEC 20 000 requires the results of the process monitoring to be recorded and reviewed to identify causes, nonconformities, and opportunities for improvement. The CSI model has been discussed in our previous paper [7].

A. *Related work*

There have been few studies which have analyzed the SLM process from the IT perspective. Jäntti and Suhonen [8] performed a research study about how to implement SLA using an ITSM tool. In their paper, Kajko-Mattsson, Ahnlund, and Lundberg [9] suggested a SLA model and evaluated it within four support organizations in Sweden. Wegman et al. [10] illustrated how methods based on the System Enterprise Architecture Methodology (SEAM) can be used to define SLA by modelling the service. Hsueh's [11] research described how an IT organization working in the aerospace industry applied an adaptive approach to ensure that service delivery meets business requirements in the face of changes in requirements. An adaptive SLM approach was used in their study to deliver a just-in-time quality service. Correia's and Abreu's [12] research work concentrated on defining and observing compliance with SLA. The main contribution of this research work was a model-based approach to SLA specification and compliance verification for IT services. Barroero's, Motta's, and Durante's [13] paper focuses on defining sustainable ways to create and manage service levels in call centres.

The purpose of this article is not to analyze successful factors of the study. There are many existing studies that have dealt with the success factors of ITSM such as the study made by Tan, Cater-Steel, and Toleman [14]. Their study focused on presenting successful factors in an Australian ITSM project. The study explained challenges and breakthroughs, confirmed a set of factors and contributed to the project's success, and offered learning opportunities to organizations.

B. *Our Contribution*

The main contributions of this paper are:

1) Defining how to implement the ITIL-based SLM practices by using the KISMET model to increase organization's efficiency.
2) Providing the lessons learned from improving SLM practices.

The results of this study can be used by persons such as service owners, service managers, process owners, process managers, and consultants, who are responsible for any phases of the IT service lifecycle. These results can be used to support CSI work based on the ITIL framework and the ISO/IEC 20 000 standard.

The rest of the paper is organized as follows. The research problem and methods are described in Section 2, and the work and results of implementing the SLM are covered in Section 3. The analysis of the findings, together with the lessons learned, is covered in Section 4. The conclusion and future work in Section 5 summarizes the case.

II. Research Methods

The research problem of this study is how to implement the ITIL-based SLM process to improve the organization's efficiency. The researchers used a case study research and action research methods with a single case organization to find answers to the research problem. The research problem was divided into the following research questions:

- RQ1: What is the current state of the SLM in the case organization (this research question is discussed in the Section III. B. and C.)?
- RQ2: What kind of things should be taken into consideration when designing SLAs and OLAs (Section III. D. provides readers with an overview of design guidelines)?

- RQ3: Which issues should be examined while analyzing whether SLAs and OLAs can be configured to the ITSM system (in Section III. E., researchers explore the ITSM system perspective of creating SLAs and OLAs)?
- RQ4: What types of benefits do SLM practices provide from the perspective of continual service improvement (the improvement cycle is visible in the whole article and relationships between CSI and SLM are discussed in the Section III. G.)?

These four research questions highlight the importance of CSI, while enhancing and deploying ITSM and SLM. This study was a qualitative research, which was built using the case study research and action research methods. According to Yin [15], a case study is ”a research strategy, which focuses on understanding the dynamics present with single settings”. During a case study, the researcher is an outsider who observes and analyses an environment and makes notes by combining different data collection methods [15]. According to Baskerville [16], the action research method produces highly relevant research results, because it is grounded in practical action, and it solves immediate problem situations. These selected methods support the situation where the researchers work together on a research project and their objective is to identify and solve problems in the IT organization's environment.

The Keys to IT Service Management Excellence Technique (KISMET) model supports action research methods, which focus on improving ITSM practices. For this reason, the researchers used the KISMET model as a tool to achieve the goals of this action research study. The KISMET model is also used in e.g., Jäntti's and Suhonen's research paper [8]. Additionally, the researchers used ITIL [1], ISO/IEC 20 000 [6], and COBIT [17] as the theoretical frameworks of this study.

A. The Case Organization

The case subject of this study was the Information System Management (ISM) unit, which is part of the IT unit of the Finnish Tax Administration. In 2012, the Finnish Tax Administration had around 5300 full-time employees from which approximately 60 work in the ISM unit. The ISM unit provided IT services (e.g., creating and maintaining user privileges, implementing changes to the software and hardware, and supporting the incidents and service requests) to these employees. The ISM unit is a representative case of a government agency with a desire to improve and enhance its IT services using ITIL-based practices.

The most of the ISM unit's employees perform service desk and customer support activities either part time or full time. The ISM unit's service desk follows ITIL-based incident management and service request management processes. In practice, this means that the number of service desk employees do not affect the support tickets handling principles, but it has influence on designing SLM agreements (defining reaction and resolution times for different types of support tickets).

B. Data Collection and Exploitation Methods

The data which was used in this research was collected by using the ITIL-based seven-step improvement process [1]. The ITIL-based seven-step improvement process consists of the following steps: 1) identify the strategy for improvement; 2) define what you will measure; 3) gather the data; 4) process the data; 5) analyse the information and data; 6) present and use the information, and 7) implement improvements.

The steps from 1 to 3 were conducted by the Finnish Tax Administration IT unit. During these steps the IT unit identified the strategy, defined metrics, and gathered the data for improving their ITSM. In Step 4, the researchers used three core perspectives of ITSM (people, process, and technology) to categorize the data that had been gathered (via a customer satisfaction survey and feedback related to resolved tickets). In Step 5, the researchers used the categorized data to identify challenges and opportunities for improving the services and processes related to ISM's practice. In Step 6, the researchers presented the ideas for improvements to the managers of the ISM unit and they made a decision to improve the SLM process. In the Step 7, the researchers implemented the improvements that had been decided by the ISM unit's on the basis of the researchers' recommendation in Step 6. This paper concentrates on the results of the Step 7, during which the researchers started the implementation of the SLM process.

The procedures of the Keys to IT Service Management Excellence Technique (KISMET) model where used to manage the SLM implementation activities. The following data collection methods and data sources were used during the research:

- **Documents and archival records:** ITSM documents, service descriptions, customer satisfaction survey, feedback data, meeting memos, and other internal records.
- **Participatory observation:** meetings and discussions with the service manager, customer manager, ITSM system specialists, and team managers from different service areas. SLM workshops held in autumn 2012.
- **Physical artifacts:** access to the intranet and to the ITSM system.

C. Data Analysis

This study performed by using within-case analysis for a single organization. According to Eisenhardt [18], the within-case method typically involves detailed case study write-ups for each site and becoming familiar with the case as a stand-alone entity. The pattern matching technique [15] was used to find patterns from the empirical data. The researchers used this technique to analyze and categorize the customer survey results and feedback according to different patterns, such as people, process, and technology.

The triangulation used in this study allowed the researchers to be more confident about their results. Denzin [19] extended the idea of triangulation beyond its conventional association with research methods and designs. During the study the researchers used three forms of triangulation [19]: 1) data triangulation, which includes collecting data through several sampling strategies; 2) investigator triangulation, which refers to the use of more than one researcher in the field to gather and interpret data, and 3) methodological triangulation, which refers to the use of more than one method for gathering data.

The process improvement events were organized into chronological order by the phases of the KISMET model. The research work was validated in weekly meetings with the ISM unit's representatives.

III. Implementation of the ITIL-based service level management process to improve organization's efficiency

The implementation of the ISM unit's SLM process was performed using the KISMET model. The model consists of the following phases: A) create a process improvement infrastructure; B) perform a process assessment; C) plan process improvement actions; D) improve / implement the process on the basis of the IT service management practices; E) deploy and introduce the process; F) evaluate the improvement of the process, and G) design continual process / service improvement actions.

A. Create a process improvement infrastructure

The "create a process improvement infrastructure" phase includes the following steps: motivate the business decision makers to ITSM, define business goals for ITSM process improvement, select an improvement target, and identify the stakeholders that participate in the process improvement.

The kickoff meeting of the SLM implementation study between the research team and the ISM unit was held in August 2012. The participants agreed that the main goal for the study was to improve and unify the ISM unit's working processes by implementing ITIL-based SLM. To achieve this goal, the research team needed to 1) evaluate the current state of the SLM in the ISM unit; 2) define SLA and OLA for the ISM unit, and 3) investigate how to configure SLAs and OLAs into the ISM unit's ITSM system.

B. Perform a process assessment

The "perform a process assessment" phase includes the following steps: perform a process assessment for a selected ITSM process, document the challenges and difficulties in the current state of the process, identify the key concepts regarding the process, study how tools support the process, and benchmark the process with ITIL best practices and ISO/IEC 20 000 requirements.

The process assessment was executed by analyzing the results (the ISM unit's internal customer satisfaction survey and feedback related to incidents and service requests that had been solved) and searching for issues and problems related to SLM. The analysis of SLM revealed following main challenges and bottlenecks: 1) there was neither knowledge nor a systematic way to perform SLM inside the ISM unit (insufficient amount of people know how to create SLAs and OLAs); 2) a common agreement between the ISM unit and customers stipulating that every incident and service request should be solved within an hour, and 3) the ISM unit needs appropriate metrics and reporting tools that could be used to improve and unify the ISM unit's working processes.

The following comments were captured from the ISM unit's customer satisfaction survey and feedback regarding SLM:

- "The delay is too long. We need help with incidents related to workstations immediately, not after a few days."
- "I don't know how long it will take till I actually get help or a solution."
- "We have been uncertain about a state or an estimated resolution time of an incident or a service request."

During the process assessment phase the researchers discovered the following strengths concerning the ISM unit's SLM. The ISM unit was interested in SLM, and both the management and personnel were strongly motivated to increase customer satisfaction and were ready to improve their ITSM system.

C. Plan process improvement action

The "plan process improvement actions" phase includes the following steps: analyze the challenges that have been identified, plan improvement actions, and validate the challenges and improvement actions.

This phase focused on defining the process improvement actions based on the challenges and bottlenecks that have been identified. For each challenge that was identified, improvement actions and the business benefit were documented.

Challenge: there is neither knowledge nor a systematic way to perform SLM inside the ISM unit (insufficient amount of people know how to create SLAs and OLAs). **Improvement actions:** the ISM unit needs to increase its knowledge of SLM methods and practices, configure their ITSM system to support SLM, and train and instruct employees to use the ITSM system efficiently. **Business benefit:** the ISM unit can define clear and measurable objectives for the SLM process. Additionally, efficient SLM can help the ISM unit to establish clear responsibilities between the ISM unit and a customer.

Challenge: there is a common agreement between the ISM unit and customers that stipulates that incidents and service requests should be solved within an hour. **Improvement actions:** the ISM unit needs to design SLAs and OLAs, which define reaction and resolution times for different types of support tickets. **Business benefit:** all incidents and service requests can be classified on the basis of their priorities. This helps employees to decide the order in which incidents and service requests should be handled, which makes support ticket handling process efficient and has direct impact to ISM unit's overall efficiency.

Challenge: the ISM unit needs appropriate metrics and reporting tools that could be used to improve and unify the ISM unit's working processes. **Improvement actions:** the ISM unit should define metrics that best meet the organization's goals. These metrics would direct the ISM unit's activities to achieve set targets. **Business benefit:** a constant monitoring allows the ISM unit to ensure that incidents and service requests are processed and solved within the agreed reaction and resolution times. Additionally, reviewing reports of reaction and resolution times allows the ISM unit to recognize weak points in processes and identify opportunities for improvement.

TABLE I. AN EXAMPLE OF SLA RULES CREATED BY SERVICE LEVEL AGREEMENT AND OPERATIONAL LEVEL AGREEMENT RULE DEFINITION MODEL

Configuration item (CI)	Type of support ticket	Priority	Reaction time	Notification of reaction time	Resolution time	Notification of resolution time	Notification targets
Software A	Incident	Low	15 min.	1 hour after	10 working days	2 working days before	Support ticket handler
		Normal	15 min.	1 hour after	3 working days	6 hours before	Support ticket handler
		High	15 min.	10 min. after	2 hours	4 hours after	Support ticket handler

D. Improve / implement the process on the basis of ITSM practices

The purpose of the ”improve / implement the process on the basis of ITSM practices” phase is to define and document: a) process goals; b) the benefits that a process provides to customers and the IT organization’s business; c) key concepts; d) roles and responsibilities; e) actions; f) metrics, and g) relationships to other ITSM processes.

The SLM workshops in the autumn of 2012 played a major role when the researchers and the ISM unit were designing SLAs and OLAs for incidents and service requests. Before the workshops were held, the research team created and sent a questionnaire to the workshop participants (e.g., the service manager, the customer manager, and ITSM specialists). Those attending were told to answer the questions from their own unit’s perspective (e.g., give an estimation of how fast the unit’s personnel could handle incidents and service requests). The questions below were included in the questionnaire:

1) Who is responsible for SLM (managing SLAs and OLAs)?
2) Which configuration items should have their own SLAs and OLAs during the first stage of the implementation of SLM?
3) Will the SLAs and OLAs for configuration items be targeted at incidents or will service requests also be taken into account?
4) What types of reaction times would be best for incidents / service requests which affect the selected configuration items (the reaction time means the time in which work should start on an incident or a service request after the ITSM system has registered the incident or the service request)?
5) What types of resolution times would be best for incidents / service requests which affect the selected configuration items (the resolution time means the time in which an incident or a service request needs to be solved)?
6) Will different priorities of service requests and incidents have an effect on reaction and resolution times? Is it necessary to define a set of reaction and resolution times for incidents and service requests on the basis of the priorities of incidents / service requests? (In this context, the priority is a category used to identify the relative importance of a support ticket. Priority is defined on the basis of the impact and urgency of the support ticket. In practice, a high priority support ticket need to solved faster than a low priority ticket.)
7) When should notification messages be sent (when the reaction or resolution time looks likely to be exceeded or has already been exceeded)? Will the priority of the incident or service request affect the sending of notifications?
8) Which person(s) or group(s) will be informed when the reaction or resolution time looks likely to be exceeded or has already been exceeded? Will the priority of an incident or service request have an effect on the sending of a notification to person(s) or group(s)?

As a result of the workshops the researcher and the ISM unit defined the SLA and OLA rules for incidents. These rules were created by using the SLA / OLA rule definition model. An example of SLA rules created by definition model is presented in Table 1.

E. Deploy and introduce the process

The ”deploy and introduce the process” phase includes the following steps: deploy an ITSM process with a pilot unit, create work instructions for how to perform the process in practice, encourage a positive attitude to ITSM among the staff, increase the awareness of ITSM in the organization through training, and organize ITSM workshops to clarify the ITSM process interfaces.

The researchers organized a workshop in September 2012 to investigate how to implement and configure SLAs and OLAs into the ITSM system. The researchers created and used the following questionnaire to evaluate the readiness of the ITSM system from the viewpoint of the implementation of SLAs and OLAs:

Creation of a new SLA / OLA

The SLM process is heavily dependent on the organization’s ITSM system. In practice, a ITSM system has to contain a SLM module (a collection of SLM features), which need to be configurable. Otherwise SLAs and OLAs cannot work properly in the ITSM system and the organization will not be able to execute SLM practices efficiently (e.g., ensure that reaction an resolution times are used as planned). The following issues should be examined while analyzing ITSM system’s principles related to the creation of new SLAs / OLAs:

- Does the ITSM system contain a proper method to create new SLAs and OLAs? Is it possible to use pre-created SLA and OLA templates?
- Is there a proper method to create a link between a SLA / OLA and a configuration item (CI)?

When a new support ticket is registered into the ITSM system, the data of the ticket will be analyzed. If the ticket contains a CI that has an existing link to SLA / OLA, this SLA / OLA will become active and the ticket needs to be resolved according to the SLA / OLA rules.

- Is there a proper method to activate a completed SLA / OLA? Is it possible to set a date when the SLA / OLA will become active?

Rule definitions for SLAs and OLAs

Every SLA / OLA contains different types of SLA and OLA rules. Defining appropriate reaction and resolution times is an essential task while designing SLA and OLA rules. A well balanced reaction and resolution times directly affect organization's capabilities to manage support tickets effectively and to keep customers satisfied. In practice, a successful creation of SLAs and OLAs requires that the attribute values of SLA and OLA rules are accurate (these values were defined in Section III. D.). The following issues should be examined while analyzing whether SLA and OLA rules can be configured to the ITSM system:

- Is there a proper method to create new SLA and OLA rules?
- Is there a proper method to edit SLA / OLA rule settings? How are the reaction and resolution time values configured into the rule?
- Is there a proper method to configure how the priority of the support ticket affects the calculation of reaction and resolution times?
- Is there a proper method to create sets of different kind of states of support tickets, which are taken into account in SLA / OLA rules?

  In practice, some of the states will affect the calculation of reaction and resolution times. For example, a state "waiting for reply from the third party" should not affect calculation of a resolution time. These kinds of states should be well-known and documented.
- Is there a proper method to create and configure working hours and holiday sets which might affect reaction and resolution times?
- Is there a proper method to configure notification message settings related to reaction and resolution times? A notification message will be send automatically to the appropriate personnel at the agreed times (e.g., when the resolution time exceeds).
- Is there a proper method to edit the content of the notification message?

Generating SLA and OLA reports

An organization has to define appropriate metrics to be able to generate accurate and relevant SLA and OLA reports. This practice helps to improve support ticket escalation and work queue management. For example, metrics can be used to measure the percentage of incidents, which have been resolved according to reaction and resolution times. This information is useful when organization reviews the functionality, validity, and business alignment of SLAs / OLAs. The following issues should be examined while analyzing ITSM system's principles related to metrics and a SLA / OLA report generation:

- Is there a proper method to configure metrics that measure how well SLAs and OLAs are working in practice?
- Is there a proper method to generate SLA and OLA reports?

At the end of the workshop, the researchers were able to determine that the ISM unit's ITSM system allows its users to create appropriate SLA and OLA rules. Based on this knowledge, the ISM unit decided that it would create SLAs and OLAs for the incident type of support tickets.

F. Evaluate the improvement of the process

The "evaluate process improvement" phase involves collecting feedback regarding an improved process, tools, and training, conducting fine-tuning if necessary, and the deployment of the processes to other organizational units or services.

After the workshop held in September, the ISM unit executed a one-month-long evaluation period. During that time the ISM unit ensured that SLAs and OLAs worked correctly in the ITSM system. This was done by: 1) creating test SLAs and OLAs; 2) testing reaction and resolution times by using different types of incidents (e.g., incidents with different priorities), and 3) checking whether notification messages got sent to the right persons at the right time.

The ISM unit was able to test the basic SLM features of the ITSM system and they confirmed that these features work correctly. However, the ISM unit also stated that one month is a too short time period to test and configure all SLM features thoroughly. The results of this evaluation period were analyzed in a workshop at the end of October 2012. Evaluate phase indicates that ISM unit has achieved the following results:

- ISM unit's managers have been able to increase their awareness related to SLM process and its purposes and practices compared the situation before research pilot (e.g., before the research only few ISM unit's managers had basic knowledge about SLM and after the research over 10 managers has now good understanding about SLM and they know how to design SLAs and OLAs).
- During the research workshops ISM unit's specialists learned how to create and configure SLAs and OLAs to the ITSM system (e.g., before the research, specialists did not have comprehensive knowledge about SLM features in the ITSM system).
- After the research, ISM unit is committed to create SLAs for the services they are providing and OLAs for different ISM unit's work groups (e.g., before the research, ISM unit used only one SLA, which covered all services).

G. Design continual process / service improvement actions

The "design continual process / service improvement actions" phase includes the following steps: conduct process reviews frequently, identify and report process improvement ideas, and plan and implement improvement actions.

The ISM unit needs to identify critical success factors (CSF), key performance indicators (KPI), and metrics for SLM in the same way as in other ITSM processes. The CSFs, the KPIs, and metrics will determine whether there are gaps between the expected outcome and the real outcome. The metrics need to be monitored and the results of the measurements should be used to identify opportunities for improvement. Ideas for improvements that are identified should be logged in the CSI register for evaluation and possible implementation. A CSI register is a database or structured document used to record and manage improvement opportunities throughout their lifecycle [1].

The metrics direct the ISM unit's activities to achieve set targets. The ISM unit has had problems with implementing the metrics that were designed because of the lack of SLM best practices. After the research described in this paper, the ISM unit is now prepared to create reports, which will help it to see how well SLAs and OLAs are working (with the possibility of handling incidents within the given reaction and resolution times) and thus, continually improve its IT services. Complete understanding of how SLM works also requires the ISM unit to measure other processes that have interfaces with the SLM process. For this reason the following CSFs and KPIs were chosen by ISM unit for incident management [5][20]:

CSF: resolve an incident as quickly as possible to minimize the impacts on the business:

- KPI: reduce the mean time required to find a resolution or a workaround for an incident, broken down by priority.
- KPI: an increased percentage of incidents resolved within the agreed resolution times by priority.

CSF: maintain user satisfaction with IT services:

- KPI: average user survey score (total and by question category).
- KPI: percentage of satisfaction surveys answered versus total number of satisfaction surveys sent.

Communication, training and documentation are required to move a new or improved service, a tool or a service management process into production [1]. The ISM unit needs to review improvement activities to ensure that approved ideas for improvement are implemented and employees use these new practices in daily basis. The ISM unit should also organize training sessions for its employees to make sure that they understand SLM practices.

IT organizations should create reports where the implemented improvement actions are presented. These reports should be delivered to employees and customers. For example, improvement actions based on customer satisfaction surveys and feedback motivate employees and customers to give feedback in the future if their input has been taken into consideration while improving the service. A report, which shows improvement trends can be used as marketing tool to communicate that the organization is committed to continual improvement [1].

IV. Analysis

In this section, the researchers analyze the research findings in the form of the lessons learned. These lessons learned can be used as general guidelines while repeating the same experience. The source for each lesson is presented using the following abbreviations: DR = documents and archival records, PO = participatory observation, and PA = physical artifacts.

Lesson I: implement a systematic way to manage and operate SLM throughout the organization (PO). Nonexistent or incoherent SLM creates different working methods and practices inside the organization and its units over a long period of time. With ITIL-based defined roles, responsibilities, processes, and metrics, the organization should be able to execute SLM in a systematic way, which improves and unifies the organization's working practices and increase organization's overall efficiency.

Lesson II: define reaction and resolution times for different types of support tickets according to the ticket's priority (PO, PA). If all support tickets are processed identically without their types and / or priorities being taken into consideration, the organization may encounter the following challenges: 1) employees who work at the service desk may have difficulties with handling support tickets at a sufficient speed, and 2) customers may also feel that they do not get solutions for their support tickets fast enough. Also, in case of high-priority support tickets, resolution time notifications should be sent after exceeding a resolution time because time limits are usually very strict and personnel do not have time to check their email messages when they are solving a ticket. In other words, these notifications should work primary as reminders to close tickets.

Lesson III: employees and customers might not have comprehensive knowledge about SLM or the benefits, which can be gained by using it (PO). If, after the successful implementation of SLAs and OLAs, a person who submits support tickets does not understand the meaning of SLAs or OLAs, he / she might not understand either why his / her low-priority support ticket takes longer to handle than high-priority tickets. For this reason, the organization needs to communicate with employees and customers about new and changed SLAs and OLAs and increase people's knowledge of SLM by organizing training sessions. These actions can be used to prevent resistance to change with regard to SLM practices.

Lesson IV: missing SLAs and OLAs might cause self-inflicted hurrying among the employees of the organization (PO). In this context, self-inflicted hurrying means that customers have unrealistic expectations about the resolution times of support tickets and employees want to handle support tickets as quickly as possible without prioritizing them first. SLAs and OLAs can be used to prevent self-inflicted hurrying among the employees of the organization because SLAs and OLAs create common rules, which both employees and customers should know and follow.

Lesson V: define appropriate metrics for SLM (DR, PO, PA). The organization needs to define and configure appropriate process metrics to gather measurement data and monitor trends and performance against service targets at planned intervals. The measurement data should be used to identify the causes of nonconformities and opportunities for improvement.

Lesson VI: organize regular reviews to evaluate how the SLAs and OLAs have been followed and report the findings to interested parties (DR, PO). It is important to define requirements for SLM reporting after the deployment of SLAs or OLAs. The requirements should answer at least the next questions: Which persons will attend to report reviews? How often will report reviews be held? What kinds of actions will be taken if the reaction and resolution times are not working properly?

V. Conclusions and Future work

The research problem of this study was this: how to implement the ITIL-based SLM process to improve the organization's efficiency. The main contribution of this study was: 1) defining how to implement the ITIL-based SLM practices by using the KISMET model to increase organization's efficiency, and 2) provide the lessons learned from improving SLM practices. In this study, the researchers used the KISMET model to improve the IT service process. The improvement focused on the following things from the viewpoint of the ITSM: evaluate the current state of the SLM in the ISM unit, define SLA and OLA for the ISM unit, and investigate how to configure SLAs and OLAs into the ISM unit's ITSM system.

There are three important reasons why our research results are valuable: First, poorly planned SLAs may cause significant financial losses in the form of sanctions when SLA rules cannot be met. Second, there are only few academic studies that deal with interface between CSI and SLM. More studies are needed to fill this knowledge gap. Third, we provided practical implications for IT organizations to enable a systematic improvement of SLM practices by using the KISMET model.

The use of case study and action research methods has certain limitations. First, the research was performed with one organization, which means that the research work needs to be repeated in other organizations, so that the results can be generalized. However, the results of this study can be used to extend ITSM theory. Other case study researchers can use the KISMET model and SLM questionnaires to get similar results while repeating this study. Second, this research was executed within a short period of time. A longer research period would have provided a more detailed analysis of how SLAs and OLAs work in practice. Third, the researchers could have conducted more SLA and OLA validation meetings with employees to get a better understanding of whether the SLA and OLA rules that were defined correspond with the reality.

More studies are needed to examine SLM and its interfaces with other ITSM processes. Further research could also focus on exploring how to assess and measure ITSM process maturity by using the ISO / IEC 15504 framework [21]. It would be also interesting to research how impacts of service improvement actions could be evaluated in IT organizations.

Acknowledgments

This paper is based on research in the KISMET project funded by the National Technology Agency, TEKES (no. 70035/10), the European Regional Development Fund (ERDF), and industrial partners.

References

[1] Cabinet Office e, *ITIL Continual Service Improvement*. The Stationery Office (TSO), United Kingdom, 2011.

[2] Cabinet Office a, *ITIL Strategy*. The Stationery Office (TSO), United Kingdom, 2011.

[3] Cabinet Office b, *ITIL Service Design*. The Stationery Office (TSO), United Kingdom, 2011.

[4] Cabinet Office c, *ITIL Service Transition*. The Stationery Office (TSO), United Kingdom, 2011.

[5] Cabinet Office d, *ITIL Service Operation*. The Stationery Office (TSO), United Kingdom, 2011.

[6] ISO / IEC, *ISO/IEC 20000-1:2011, IT Service management, Part 1: Service management system requirements*. ISO/IEC JTC 1/SC 7, 2011.

[7] S. Heikkinen and M. Jäntti, "Establishing a continual service improvement model: A case study," in *Proceedings of the 19th European Conference: Systems, Software and Service Process Improvement (EuroSPI)*, 2012, pp. 61 – 72.

[8] M. Jäntti and A. Suhonen, "Improving service level management practices: A case study in an IT service provider organization," in *International Conference on Advanced Applied Informatics (IIAIAAI)*, 2012, pp. 139 –144.

[9] M. Kajko-Mattsson, C. Ahnlund, and E. Lundberg, "Cm3: service level agreement," in *Proceedings of 20th IEEE International Conference on Software Maintenance*, 2004, pp. 432–436.

[10] A. Wegmann, G. Regev, G.-A. Garret, and F. Marechal, "Specifying services for ITIL service management," in *International Workshop on Service-Oriented Computing Consequences for Engineering Requirements (SOCCER '08)*, 2008, pp. 8 –14.

[11] M.-C. Hsueh, "Adaptive service level management," in *11th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, 2007, pp. 451 – 456.

[12] A. Correia and F. B. e Abreu, "Defining and observing the compliance of service level agreements: A model driven approach," in *Seventh International Conference on the Quality of Information and Communications Technology (QUATIC)*, 2010, pp. 165 –170.

[13] T. Barroero, G. Motta, and M. Durante, "Sustainable service level agreements," in *IEEE International Conference on Services Computing (SCC)*, 2011, pp. 679 – 684.

[14] W.-G. Tan, A. Cater-Steel, and M. Toleman, "Implementing IT service management: A case study focussing on critical success factors," *Journal of Computer Information Systems*, vol. 50, no. 2, 2009.

[15] R. K. Yin, *Case Study Research: Design and Methods*. CA: Sage Publications, 2003.

[16] R. L. Baskerville, "Investigating information systems with action research," 1999.

[17] COBIT 4.1, *Control Objectives for Information and related Technology: COBIT 4.1*. IT Governance Institute, 2007.

[18] K. M. Eisenhardt, "Building theories from case study research," in *Academy of Management Review*, 1989, pp. 532–550.

[19] N. Denzin, *The Research Act in Sociology*, 1970.

[20] Office of Government Commerce (OGC), *Planning to implement service management*. The Stationery Office, Norwich, 2002.

[21] Public Research Centre Henri Tudor, *ITSM Process Assesment Supporting ITIL*. Van Haren Publishing, Zaltbommel, 2009.

Measuring the Functional Size of Real-Time and Embedded Software: a Comparison of Function Point Analysis and COSMIC

Luigi Lavazza and Sandro Morasca
Dipartimento di Scienze Teoriche e Applicate
Università degli Studi dell'Insubria
Varese, Italy
{luigi.lavazza; sandro.morasca}@uninsubria.it

***Abstract*— The most widely used methods and tools for estimating the cost of software development require that the functional size of the program to be developed be measured, either in "traditional" Function Points or in COSMIC Function Points. The latter were proposed to solve some shortcomings of the former, including not being well suited for representing the functionality of real-time and embedded software. However, little evidence exists to support the claim that COSMIC Function Points are better suited than traditional Function Points for the measurement of real-time and embedded applications. Our goal is to compare how well the two methods can be used in functional measurement of real-time and embedded systems. We applied both measurement methods to a number of situations that occur quite often in real-time and embedded software. Our results seem to indicate that, overall, COSMIC Function Points are better suited than traditional Function Points for measuring characteristic features of real-time and embedded systems. Our results also provide practitioners with useful indications about the pros and cons of functional size measurement methods when confronted with specific features of real-time and embedded software.**

Keywords- Functional Size Measurement; Function Point Analysis; COSMIC Function Points; Real-time software; Embedded software

I. INTRODUCTION

Several methods have been proposed to estimate the development effort of a software product, given the characteristics of the product itself and its development process. Software size plays a special role in effort estimation, as it is the main input used by the vast majority of effort estimation models. Accordingly, measures of *functional* size are used in early effort estimation models, since other measures –like Lines of Code– are not available in the early development phases. Functional measures quantify the functional size of a software application, as defined in the requirements specification documents.

The available functional sizing methods are evolutions of the Function Points Analysis (FPA), originally proposed by Allan Albrecht [1]. The International Function Points User Group (IFPUG) maintains the definition of the method and publishes and regularly updates the official Function Point (FP) counting manual [2][3]. Effort estimation methods have been defined, and tools supporting them have been developed, which require the size in FP as the main input.

FP are generally not considered well suited for measuring the functional size of embedded applications. The reported motivation is that FP –conceived by Albrecht when the programs to be sized were mostly Electronic Data Processing applications– capture well the functional sizes of data storage and data movement operations, but are ill-suited for representing the complexity of control and elaboration that are typical of embedded and real-time software.

The COSMIC method was defined to overcome some limitations of FPA. The COSMIC method [4] redefines FPA's basic principles of functional size measurement in a way that applies equally well to traditional "business" application and other applications, including the real-time and embedded ones. Specifically, the COSMIC method counts the data movements (entries, exits, reads and writes) that involve data groups (corresponding approximately to FPA's logic files) in each functional process (corresponding to FPA's elementary processes). The result is a functional size measure called COSMIC Function Points (CFP).

Even though it is traditionally considered not well suited for real-time and embedded applications, FPA can be applied to embedded software via a careful interpretation of FP counting rules [5]. Moreover, it is known that many real-time projects have actually been measured using FPA. On the contrary, there is little *analytic* evidence of successful applications of the COSMIC method to real-time and embedded applications. This paper aims at providing some evidence about the suitability of FPA and the COSMIC method to measure real-time embedded software.

Both FPA and COSMIC methods require the representation of user requirements according to a method-specific model of software (e.g., the FP model includes logic files and elementary processes, while the COSMIC model includes functional processes and data movements). Measurement is then based on counting the elements of these models according to given rules. To measure RT and embedded software, it is of critical importance that representative models can be correctly derived from the user requirements. To test this ability, we consider a set of typical and representative –though necessarily incomplete– features of real-time embedded software and apply FPA and COSMIC to each of them. The comparison of the two methods provides useful indications to the developers that have to choose a functional size measurement method.

The paper is organized as follows: Section II illustrates the attractiveness of the COSMIC method from the management point of view. Section III presents a set of modeling and measurement problems that occur frequently in real-time and embedded software developments. In Section IV, FPA and COSMIC methods are applied to the cases illustrated in Section III. Section V accounts for related work, while Section VI draws some conclusions and outlines future work.

Throughout the paper, we refer exclusively to Unadjusted Function Points (UFP) for FPA, because UFP are more commonly used than adjusted Function Points and because UFP are recognized as an ISO standard, while FP are not.

II. Sizing and Estimation of Real–Time Embedded Software: the Manager's Point of View

Both FPA and COSMIC methods aim at measuring the size of Functional User Requirements (FUR). However, there are a few reasons that suggest that the COSMIC method may be preferable. First, CFP are defined in a simple and sound way, while the definition of FP has been widely criticized, e.g., because the weighting mechanism make unclear whether FP are a measure of size or effort [6], or because the inherent subjectivity of FPA leads even certified measurers to measure different sizes for the same application [7][8]. Finally, the COSMIC method, which does not require a thorough analysis of data and allows for analyzing transactions at coarser granularity level, is somewhat faster and less expensive than FPA.

So, managers have a few reasons to prefer the COSMIC method over FPA. However, evidence concerning the suitability of the COSMIC method for measuring real-time software is still missing. This paper aims at filling this gap.

III. Case Studies for Functional Size Measurement of Real-Time Embedded Software

Here, we illustrate a set of typical features of real-time and embedded software that are difficult to represent by means of the models that underlie the definition of functional size measurement methods. All the proposed cases are derived from the first author's experience gained in measuring seven avionics applications in a large European company. So, the proposed set of cases is of empirical origin: during the measurement, the cases presented here emerged as those particularly challenging for functional size measurement. Most examples are illustrated by means of sequence diagrams, according to the measurement-oriented modeling methodology proposed in [9] and used in [10]. It is assumed that the reader is familiar with FPA and COSMIC concepts and terminology and with UML.

A. *Embedded processes having multiple purposes*

In embedded software, several processes often include both updating some data and producing some result. Consider for instance a process that initializes and tests a piece of hardware (Fig. 1): both the initialization and the test are necessary. Actually, the initialization and test of several hardware devices are performed by means of a single command: you send the initialization command and get the resulting state back, so that you can check that the device is working correctly.

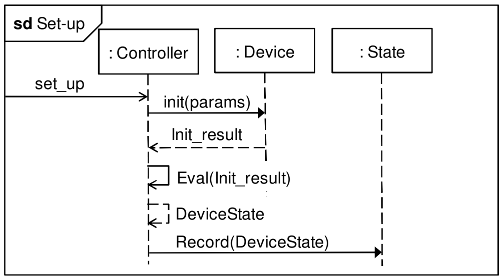


Figure 1. Inizialization of devices: the "main purpose" is not evident.

B. *Transactions defined at very low level*

Requirements often concern very low level operations, thus making it difficult to identify functions that match the definition of Base Functional Components.

1) Memory vs. data

In embedded software, the use of RAM as a whole introduces new requirements. For example, a piece of software embedded on board of a military airplane should clear the whole RAM under given circumstances, e.g., if the airplane crashes in an enemy zone (because the information stored in memory must not be made available to enemies). This requirement (Fig. 2) is peculiar in that it is about the whole RAM, not the user-relevant data.

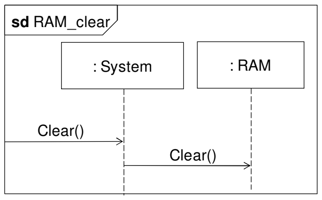


Figure 2. RAM clearing process.

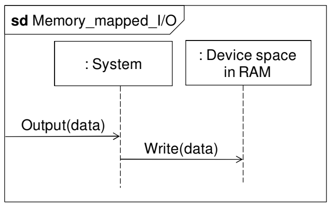


Figure 3. Memory-mapped I/O.

2) Memory mapped I/O

In embedded systems, updating a variable and sending data to a device can be extremely similar operations. For instance, when I/O is memory-mapped, both mentioned operations write registers or RAM locations (Fig. 3).

3) Processes that do not terminate properly

In embedded software, it is often required that a function terminates by jumping to a given location. This situation is illustrated in Fig. 4: the initialization function terminates by executing the set-up function (described in Fig. 10).

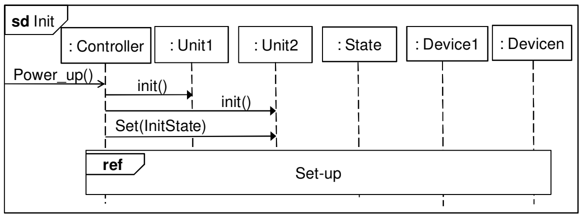


Figure 4. A function that ends with a jump to another function.

C. *Taking into account the devices*

In traditional software applications, functions are usually invoked by the user and end either by updating some internal data, or by outputting some information. In embedded applications, the situation can be very different. Often it is some hardware device (not a user) that acts as both the cause that determines the execution of the function and the destination of the produced data or signals.

1) Considering the role of the Operating System in I/O

Let us consider the following requirements for an I/O functionality (described in Fig. 5): “upon request by the controller, data are retrieved from an I/O channel, according to the criteria stored in the I/O channel table. When all the data have been read, they are suitably converted and sent back to the controller.” It is often the case that the I/O operation has to be carried out with the help of the Operating System and the requirements can be implemented by means of two functions, illustrated in Fig. 6 and Fig. 7. The first function (Fig. 6) is invoked by the controller and prepares an I/O request for the OS and a subsequent system call. The second function (Fig. 7) is triggered by the interrupt from the I/O device and involves reading the data from the channel, elaborating them, and sending them back to the controller. The execution of this “function” is done partly by the OS (by a driver that will have to be implemented as a part of the application development) and partly in the section of the application devoted to I/O.

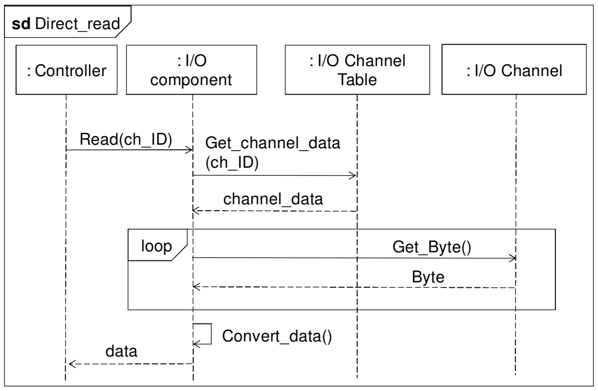


Figure 5. Process featuring direct access to I/O channels.

If the development also includes the construction of a driver for the considered I/O device, it seems that taking into account the size of the corresponding code will contribute to produce a more accurate effort estimate. In other words, it seems reasonable to count two functions, corresponding to the “elementary processes” described in Fig. 6 and Fig. 7.

2) Multi cycle operations

In real-time systems, it is not unusual that a function is too long to fit into one execution cycle. In such cases, it is rather common to split the function into two (or more) pieces that are executed in consecutive execution cycles. Here are two typical examples:

- The function transfers data via a buffer. The data to be transferred do not fit in the buffer. The transfer is split into n cycles: in each cycle 1/n of the data are copied into the buffer.
- The function, triggered by the tick, takes a time longer than the cycle duration (i.e., the time between two consecutive ticks) to execute. Thus, the transfer is split into multiple consecutive cycles.

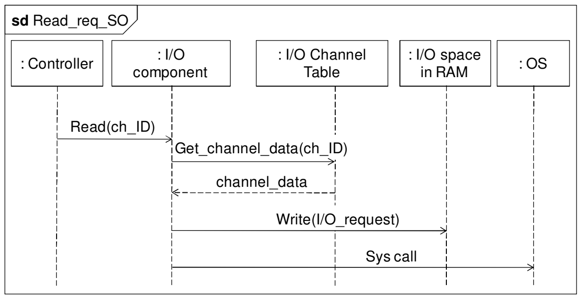


Figure 6. Process Access to I/O channels via the O.S.

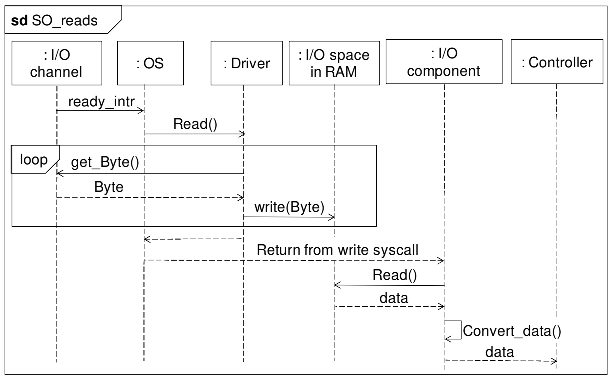


Figure 7. The O.S. handles the I/O.

An example is given in Fig. 8: an output operation is split over two consecutive clock cycles. In the first cycle the application outputs the data from Data_1 and sets the State to represent that there is a pending output operation; in the following cycle, the State indicates that the output operation has to be completed, thus data are read from Data_2 and sent to the output device.

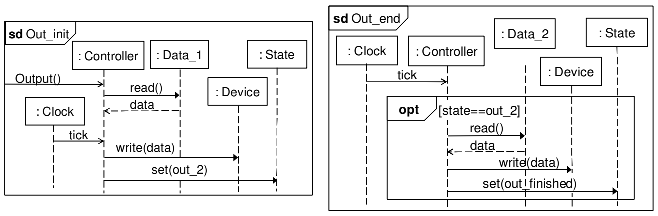


Figure 8. Output: first and second (final) cycle.

These cases are often described in the requirements, since they deal with the real-time behavior of the application, which is typically explicitly accounted for in the requirements specification.

However, requirements specifications could not state explicitly that the function should be split, i.e., requirements could just describe the whole operation as in Fig. 9.

D. *Long processes*

In embedded software, functions are often “service routines” that perform rather long tasks; e.g., the requirements specify that “the connected devices are tested, and the result (a ‘pass’ value or the set of diagnostics) is sent to the controller, which stores it for later use.” Fig. 10 illustrates the situation with 4 different device types.

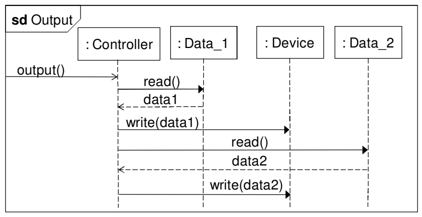


Figure 9. Output, not split.

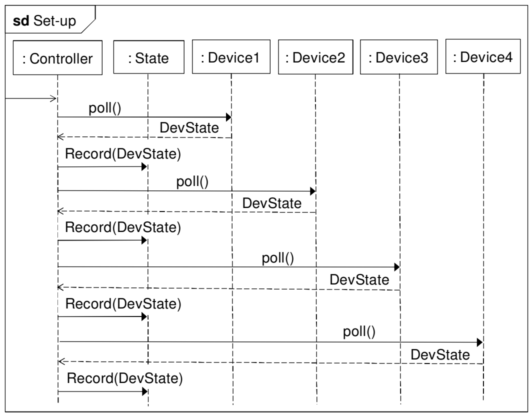


Figure 10. A long transaction.

E. *Unusual data*

Embedded applications often include constant data structures (e.g., data mapping tables or bit masks) that require a non-negligible design effort, which we would like to take into account. An example is shown in Fig. 5: for each request to read an I/O channel, the I/O component reads from the channel table how many bytes must be read from the channel and how they should be interpreted. The channel table is a read-only structure that describes how to manage the I/O channels.

F. *Complex elaborations*

In real-time and embedded applications, some operations can be complex. Consider for instance the generic flight control operations described in Fig. 11. It should not be surprising that the computation of the flight control data can be quite complex.

IV. Applying FPA and COSMIC to Real-Time Embedded Software

This section illustrates the application of FPA and COSMIC methods to the cases described in Section III.

A. *Embedded processes having multiple purposes*

According to the IFPUG counting rules [2][3], the size of a function varies according to its type (external input, output or query). The type is determined by the "main purpose" of the function, according to the requirements. However, it may be difficult to decide what the main purpose is, since both the external input and the external output can update internal data and report a result, as in our case. In conclusion, measures based on FPA have some degree of subjectivity that can be hardly avoided.

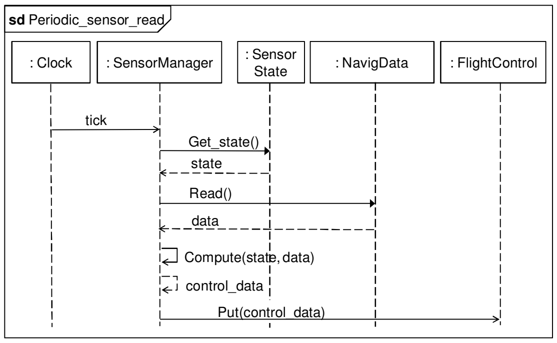


Figure 11. Sensor-driven flight control.

The problem described above does not apply to COSMIC measurement, since all processes are treated in the same way, regardless of their purpose.

B. *Transactions defined at very low level*

1) *Memory vs. data*

According to the principles of FPA, in a case like the one described in Section III.B.1) one should count the memory clearing function as an external input. In that case, since every External Input (EI) manages an Internal Logic File (ILF), we should consider the RAM an ILF. On the one hand, counting the RAM as an ILF does not appear correct with respect to the rules, since logic data files should represent a homogeneous set of related data (which RAM is not), on the other hand, not considering the RAM as an ILF is an inconsistency, as all EI have to deal with an ILF.

There is a similar problem with the COSMIC method, as the process writes in the RAM: accordingly, we should consider a write data movement. However, this implies that the RAM is classified as a data group, which does not appear perfectly coherent with the COSMIC rules.

2) *Memory mapped I/O*

When I/O is memory-mapped, an output operation can be modeled as an External Output (EO) but also as an EI since the output is obtained by writing registers or RAM locations (see Fig. 3). The choice affects the resulting measure, since EI and EO have different weights. With the COSMIC method, you still can model the operation as a Write or an Exit data movement, but the choice does not affect the final measure, since every data movement contributes exactly one CFP.

3) *Processes that do not finish properly*

According to FPA, a transaction function has to be self-contained and leave the application being counted in a consistent state. In embedded software, it is often required that a function terminates by jumping to a given location (Fig. 4). In this case, the transaction is not self-contained and does not leave the program in a consistent state. FPA does not suggest how to take into consideration this type of functions. Just ignoring them would not be a good idea, since it takes some effort to implement these functions; hence we want them to contribute to the functional size of the application. Actually, there is no other way of dealing with these cases than just ignoring the constraints imposed by the IFPUG and counting the functions, considering their behavior down to the final jump. The same problem occurs when the COSMIC method is used, since functional processes are defined as FPA transactions, in essence.

C. *Taking into account the devices*

1) *Considering the role of the Operating System in I/O*

With both FPA and COSMIC methods, the measurement of the process represented in Fig. 5 is quite straightforward. The problem here occurs when the development must also include the construction of a driver for the considered I/O device, since taking into account the size of the corresponding code will contribute to produce a more accurate effort estimate. In other words, it seems reasonable to count two functions, described in Fig. 6 and Fig. 7.

With FPA, this requires a deviation from the FPA counting practice, since FPA does not take into account the existence of different "layers": with FPA you can only measure requirements at the single abstraction level corresponding to the user's point of view, and the user is not aware of the OS and what happens in the OS.

With the COSMIC method, it is possible to explicitly model and measure the layers that compose the software application. The sum of the sizes of the layers is generally greater than the size of the whole application corresponding to the point of view of the user (who is not aware of the existence of layers). So, the measure of layers is exactly what is needed to take into account the size of the OS parts that are being developed.

2) *Multi cycle operations*

The cases described in Section III.C.2) suggest that the value of a functional size measure can depend on how requirements are written. Let us consider the case when requirements specifications do not state explicitly that the function should be split (Fig. 9): if Data_1 and Data_2 account for 10 DET each, the transaction is a high complexity EO (having 3 FTR and 21 DET), whose size is 7 FP. When requirements specifications prescribe that the function be split (Fig. 8) we have two average complexity EO (3 FTR and around 12 DET each), whose size is 10 FP in total. When requirements specifications do not state explicitly that the function should be split, the COSMIC method identifies one functional process sized 5 CFP, since it involves 5 data movements (the Entry, the Reads of Data_1 and Data_2, and the corresponding Exits). When requirements specifications prescribe that the function be split, according to the COSMIC rules we have two functional processes, one involving 5 data movements (the Entry that triggers the operation, the Read of Data_1, the Entry of the clock tick, the Exit to the device, the Write of the state), and one involving 4 data movements (the Entry of the tick, the Read of Data_2, the Exit to the device, the Write of the state); the total size is thus 9 CFP.

In conclusion, both methods provide measures of size that depend on how requirements are written. This is a characteristic of the methods that has to be taken into account, as it affects the resulting measures.

D. *Long processes*

A well known problem with Function Points is the so-called "cut-off" effect: a function cannot contribute more than 7 FP to the functional size, regardless how many DETs it moves and how many FTRs it involves. This is a relevant problem, especially in embedded software, where functions are often "service routines" that perform rather long tasks, like in the example illustrated in Section III.D and Fig. 10.

Fig. 10 illustrates the situation with 4 different device types. According to the IFPUG counting rules, this is a single transaction. If the device states contain on average 5 (or more) parameters, then the transaction is a complex one. The problem here is that if we had 5 or more different types of devices, the number of FP would not increase with the number of devices: according to FPA, we would have just one complex EI. This is a problem, because in practice the development effort increases with the number of device types, since each device type provides different status data, which need to be interpreted in a specific way.

FPA hides from the estimation methods how much a function is bigger (thus more expensive to build) than another that classifies as complex. The COSMIC method, on the contrary, does not suffer from the cut-off effect. In a case like the one in Section III.D and Fig. 10, the size in CFP takes into account *all* the data movement, whose number is proportional to the number of devices.

E. *Unusual data*

According to FPA, data functions are either internal data "maintained" (i.e., modified) by the application, or external data (maintained outside the application). Constant data are treated as "decoding data" and explicitly excluded from the counting [2]. However, it seems that the authors of the IFPUG manual had in mind simple "zero effort" constants when they wrote the rules concerning the constant data.

To account for the fact that a constant data structure will require some design effort, it is necessary to deviate from the IFPUG rules, and count a "constant ILF": for instance, in the example illustrated in Fig. 6, one should count an ILF for the channel table; consistently, a FTR for each access to the table should be considered.

The COSMIC method does not count data directly; that is, no fraction of the size measures accounts for data. On the contrary, data movements are counted without considering whether the data being moved are constant or not. In conclusion, this case does not pose any additional difficulty to the application of the COSMIC method.

F. *Complex elaborations*

Both FPA and COSMIC methods base the measurement of size on the number of processes and the amount of data handled. For instance, the process described in Fig. 11 is considered as an EO (with a maximum size of 7 FP) or a functional process accounting for 4 CFP (as it involves 4 data movements). None of the two methods considers the complexity of the computations performed: the fact that the "Compute" operation performed in the process is simple or complex does not change the size of the process.

This is clearly a shortcoming of the two methods, since the development effort is very likely proportional to the complexity of the functions to be implemented.

V. Related Work

There is a fairly large body of literature aimed at extending the scope of functional size measurement to real-time software. Mark II Function Points [11][12] refine and extend the traditional function point transaction model and environmental factors. Asset-R [13] extends the applicability of FP to real-time systems by considering issues like concurrency, synchronization, and reuse. It also accounts for architectural, language expansion, and technology factors to generate the size estimate. Application Features [14] aim at the early estimation of the size of application in the process control domain. Counting practices for highly constrained systems [15] address issues such as boundary identification and internal processing. Also the IFPUG published a Case Study on how to apply FPA to real-time software [16].

A common characteristic of the methods mentioned above is that none of them is widely used in practice. A partial exception is represented by Mark II Function Points [11], which were also standardized [12]. So, the popularity of FPA and COSMIC suggested that their suitability to deal with software has to be evaluated.

VI. Conclusions

The results of our analysis show (see Table I) that several cases can be measured with the COSMIC method by just applying the measurement rules given in the manual [4], while Function Point Analysis often requires "bending" the rules to account for the considered cases. Also the resulting measures are easily affected by the measurement choices made in FPA, while there are just a few cases (namely, processes terminating with a jump, multi-cycle operations and complex elaborations) that can affect the measures in CFP.

TABLE I. Comparison of FSM methods

Case	FPA		COSMIC	
	Rules	*Meas.*	*Rules*	*Meas.*
Multiple purpose processes	✗	✗	✓	✓
Memory data	✗	✗	✓	✓
Memory mapped I/O	✗	✗	✗	✓
Processes terminating with jump	✗	✗	✗	✗
Clock	✓	✓	✓	✓
OS involved in I/O	✗	✗	✓	✓
Multi cycle operations	✓	✗[a]	✓	✗[a]
Long processes	✗	✗	✓	✓
Unusual data	✗	✗	✓	✓
Complex elaborations	✗[b]	✗	✗	✗[b]

[a] The measures depend on how requirements are written.
[b] Elaboration complexity is just not accounted for by any rule.

In conclusion, the original claims that the COSMIC method is more suitable than FPA for measuring real-time and embedded applications seem justified.

In any case, it must be noted that neither FPA nor the COSMIC method account for the complexity of the required elaboration. This may be a problem in the real-time embedded context, since some processes can be really very complex and require a relevant amount of development effort. Future work involves assessing measures that represent not only the functional size of Real-Time applications as done by FPA and COSMIC methods, but can represent also the complexity of the required elaboration.

References

[1] A.J. Albrecht, Measuring Application Development Productivity, Joint SHARE/ GUIDE/IBM Application Development Symposium, 1979, pp. 83-92.

[2] International Function Point Users Group. Function Point Counting Practices Manual - Release 4.3.1, January 2010.

[3] ISO/IEC 20926: 2003, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual, Geneva: ISO, 2003.

[4] COSMIC – Common Software Measurement International Consortium, The COSMIC Functional Size Measurement Method - version 3.0.1 Measurement Manual, May 2009.

[5] L. Lavazza and C. Garavaglia, "Using Function Points to Measure and Estimate Real-Time and Embedded Software: Experiences and Guidelines", ESEM 2009, Lake Buena Vista, FL, USA, October 15-16, 2009, IEEE, pp. 100-110.

[6] A. Abran and P.N. Robillard "Function points: a study of their measurement processes and scale transformations", Journal of Systems and Software, vol.25,n.2, Elsevier, 1994, pp.171-184.

[7] C. Kemerer, "Reliability of Function Points Measurement: a Field Experiment," Comm. ACM, Vol. 36, No. 2, 1993, pp. 85-97.

[8] J.R. Jeffery, G.C. Low, and M.A Barnes, "Comparison of Function Point Counting Techniques," IEEE Trans. Software Eng., Vol. 19, No. 5, 1993, pp. 529-532.

[9] L. Lavazza, V. del Bianco, and C. Garavaglia, "Model-based Functional Size Measurement", 2nd Int. Symp. on Empirical Software Engineering and Measurement – ESEM 2008, Kaiserslautern, Germany. October 9-10, 2008, pp. 100-109.

[10] L. Lavazza and V. del Bianco, "A Case Study in COSMIC Functional Size Measurement: the Rice Cooker Revisited", IWSM 2009, Amsterdam, November 2009, pp. 101-121.

[11] C.R. Symons, "Function Point Analysis: Difficulties and Improvements", IEEE Transactions on Software Engineering, Vol. 14, No. 1, January, 1988, pp. 2-11.

[12] ISO/IEC 20968: 2002, Software engineering Mk II Function Point Analysis. Counting Practices Manual, International Standardization Organization, ISO, Genève, 2002.

[13] D. J. Reifer, "Asset-R: A Function Point Sizing Tool for Scientific and Real-Time Systems", Journal of Systems and Software, Vol. 11, No. 3, March 1990, pp. 159-171.

[14] T. Mukhopadhyay and S. Kekre, "Software Effort Models for Early Estimation of Process Control Applications", IEEE Transactions on Software Engineering, Vol. 18, No. 10, October 1992, pp. 915-924.

[15] European Function Point Users Group, Function Point Counting Practices for Highly Constrained Systems, 1993.

[16] IFPUG, Case Study 4: Counts Function Points for a Traffic Control System with Real Time Components, International Function Point Users Group – IFPUG.

MCReF: A Metric to Evaluate Complexity of Functional Requirements

Carlos Roberto Paviotti
São Paulo Federal Institute of Education, Science and Technology, IFSP
Capivari, Brazil
e-mail: carlinhos@ifsp.edu.br

Luiz Eduardo Galvão Martins
Institute of Science and Technology
Federal University of São Paulo, UNIFESP
São José dos Campos, Brazil
e-mail: legmartins@unifesp.br

Abstract— **The high sophistication of software systems has lead to an increase in the requirements complexity. Currently, there are metrics to evaluate the functional size of the software such as metrics of function point and use case points which are used with good results. However, a metric for the complexity for software requirements specifically had not yet been proposed. Identifying this gap, this paper proposes a Metric of Complexity of Functional Requirements (MCReF is an acronym composed by Portuguese words: *Métrica de Complexidade de Requisitos Funcionais*) indicated to evaluate and classify the complexity of software requirements. MCReF was developed from an empirical study based on a questionnaire that collected the opinion of 20 professionals from the requirements area to determine the weights of the factors that influence the requirement complexity. The responses were tabulated and given a statistical treatment to assess the weights of the complexity factors and their respective ranges of values for classification. A case study using MCReF is also presented in this paper.**

Keywords-Requirements Engineering; Complexity of Requirements; Requirement Metrics.

I. INTRODUCTION

Being part of the system engineering phases, Requirements Engineering consists of a set of techniques employed in the processes involved in the development of system requirements, i.e., eliciting, detailing, documentation and validation of the requirements [11]. The result of the set of requirements is a Software Requirements Specification Document, where the degree of understanding and accuracy of the provided description tend to be proportional to the degree of quality of the generated product. The definition of the software requirements occurs in the early development phases. Requirements Engineering provides methods, techniques and tools that help requirements engineers to define and classify what must be implemented in the software before starting building the system to be, i.e., the earliest phases of the software life cycle. Several processes models advocate such a procedure, for example: Requirements Definition in the Waterfall Model [13], Requirements Design in the Spiral Model [11], Requirements Gathering in the Prototyping Model [15], Requirements Workflow in USDP [13], etc. Among the ways of realizing the requirements complexity of a given system, regardless of the process model to be adopted, the Use Cases provide help in this issue, helping to formalize the scope of the system and facilitating the communication between developer teams and stakeholders. The presentation of requirements in a Use Cases Diagram is a simplified and less complex form of representation than the requirements description in natural language, enabling to estimate the project size and realize the system's complexity in a global way. Being one of the important factors to generate a software product with quality, a Software Metric corresponds to quantitative measures on one or more relevant features of the software [7][8][10], which allows developers to have a more refined view on the software process or related documentation, along with being an important management tool that contributes to preparation of time schedule, more accurate costs and more plausible goals, thus facilitating the decision making process and its consequent results.

Among the existing metrics, focusing on functionalities and not on a software system requirements, there are Function Points [13] and Use Cases Points Metrics [15], in both, the specified complexity factors are classified as subjective since they link the measures to "its value to the user".

Some related studies have been performed involving the requirements complexity, with presence in researches and empirical studies [12]. However, as many of them are focused on software quality, the necessity of involving the complexity factor in achieving the final result of the study remains, which generally refers to the system or project complexity in relation to their functionalities and not their requirements.

Kanjilal, Sengupta, and Bhattacharya [1] developed an approach based on metric model which aims to quantitatively estimate the requirements complexity for the object-oriented methodology, using project models like Sequence Diagram and Classes Diagram in the aid of validating the estimates in the project phases and long term project management.

Zhao, Tan, and Zhang [2] created a method to estimate costs through the requirements designing, proposing a new term named Path Complexity, which indicates a metric to measure the effort of the software complexity based on E-R Diagram (Entity-Relationship Diagram), showing the whole database structure in which an entity that can reach other entities due to its relationship and obtaining data on it.

Aiming the complexity related to requirements, an empirical study performed by Regnell, Svensson, and Wnuk [3] describes a case of system engineering in the field of mobile telephony, based on experiences used at Sony

Ericsson, which demonstrates the existing complexity of requirements in mobile telephones development.

The result of this study is called by the authors Very Large–Scale Requirements Engineering (VLSRE), suggesting a new order of magnitude applied to requirements, focusing on the size of the requirements set (the number of requirements is used, among other variables, to represent the complexity and it is strongly related to the nature of interdependencies among requirements), which are managed by a system developer company.

Complexity is an attribute that allows measuring if a software, usually part of it (module, method or function) is easy to read (comprehension), or else how complex it can become, if it contains a large number of nesting of laces and decision commands in a given program or functionality [8].

According to McCabe (1976) in Pressman [13] complexity is the quantification of the number of interdependent paths in a program, which provides an indication of its maintainability and testability. It is important to note that these definitions of complexity were built with the software as object in question and not the software requirements [10]. Another issue, also reported by Regnell, Svensson, and Wnuk [3] is that one of the factors responsible for the increasing of the requirements complexity is the large and diversified set of stakeholders, both internal and external to the organization. Based on the research performed in the literature and on the case studies, it is possible to characterize the requirements complexity as the degree of difficulty to interpret, specify, understand and implement a set of requirements, which is directly influenced by the amount of variables and procedures relevant to the requirements, as well as by the dependency relationships or coupling among them.

Currently, there is not available among the Requirements Engineering techniques, a metric aimed specifically to evaluate the requirements complexity. Such metric is of fundamental importance for the software development teams to have a reference concerning to the degree of complexity a requirement can present. Based on a metric of requirements complexity, the developer teams may build their own productivity indicators, which will be of great value to accurately estimate variables such as effort, time and cost of software development.

The aim of this study is to contribute to the software development in industries that employs the Requirements Engineering concepts and techniques, by proposing a metric to evaluate the complexity of functional requirements, even before start building the systems, in which this complexity is already recognized in the early phases of the life cycle of the software development.

To achieve the proposed metric, the adopted methodology was divided in four phases: (i) Development of case studies focusing the requirements elicitation, specification and validation, based on real contexts, including: a) Creation of a requirements specification document using the template *Volere*, referring to a system for monitoring and capturing heart rates to evaluate the heart autonomic function (in human beings); b) Creation of a requirements specification document using one of the templates from the *IEEE STD 830-1998* recommendation [9], regarding to the system for technical and physical monitoring of athletes in all the categories of a Brazilian professional soccer club [16]. These case studies were used as a "laboratory" to identify the factors that influence the requirements complexity. (ii) Creation of a Requirements Complexity Metric, identifying: a) main variables that influence the requirements complexity; b) Relationships among these variables; c) Weight of these variables, obtained through the application of a questionnaire to the software development professionals; d) Classification of the requirements complexity. (iii) Application of the proposed metric in three case studies which were software projects whose requirements had already been raised and previously documented. (iv) Analysis and discussion of the results obtained with the application of the metric in the case studies.

The rest of this paper is organized as follows: MCReF metric is explained in the section II. The empirical study that grounded the proposed metric is presented in section III. A case study is discussed in the section IV. Conclusions are presented in the section V.

II. MCREF METRIC

A. *Proposal*

The revolution of software systems, where the increasing complexity and the size of their set of requirements are inherited factors of this progress, has motivated the improvement of already existing methods, techniques and tools in the Requirements Engineering.

Currently, there are metrics to estimate the software size and functionality [8][13][15], something that was a challenge to software companies in past decades. However, a metric for complexity of software requirements had not already been proposed. Motivated by such a gap, this paper presents the Metric of Complexity of Functional Requirements (MCReF).

MCReF is a metric proposed to evaluate the complexity of functional requirements, enabling to classify how complex is the functional requirement, focusing especially in information systems requirements. To apply the proposed metric it is necessary to obtain from the Requirements Specification Document, the generated artifacts or diagram, enabling to know the main factors that influence the complexity of functional requirements, namely: treatment and identification of functionalities, input and output variables, dependencies and couplings, decompositions, constraints and number of stakeholders involved in. Once performed the identification of these factors, it is necessary to specify them a little more, and thus to classify the sub-factors that influence the complexity of functional requirements on which is applied the weight attributed to each subfactor of complexity, enabling to obtain the degree of complexity in a single requirement.

B. *Case study Development*

To assist identifying the factors that influence the complexity of the information system requirements, two case studies were carried out, each one having as a result a requirements specification document, being in different templates, which allowed a wider view of the functionalities and the objectives to specify and document the requirement correctly. The requirements specification documents included the following systems: (i) Monitoring and Heart Frequency Capturing System to evaluate the heart autonomic function (in human beings) developed in collaboration with Department of Physiotherapy at UNIMEP (Methodist University of Piracicaba – Brazil), using Volere template [14]; (ii) Technical and Physical Follow Up System to all categories of a professional soccer club in Brazil, which is discussed in a previous work [16].

C. *Metric Development*

Based on case studies performed to support the MCReF metric it was possible to identify in the Requirement Specification Document [16], the main factors of the complexity that influence the functional requirements, which are described in the following subsections.

1) *Input and Output Variables*

Represent values to be treated or used to meet the requirement represented by the identifiers, i.e., a label for each variable. They are classified as: (i) Input variable – existing variable in the requirement that will receive information from one agent or another system and making necessary to treat the value of this input, for example, an input variable of genre: "f" for female or "m" for male. (ii) Output variable – a variable of the result of the requirement. After processing the variable, the resulting information will be presented to the applicant and such value must be treated by the application, for example: the information "f" obtained from a field that stores data referring to genre must present the result "female" to the user requesting. It is possible to identify this factor of complexity in the Requirement Specification Document due to: the large number of variables, which will possibly have a greater complexity when comparing to requirements with a few variables, because these, whether input or output, need to be treated to present the results they were intended; the amount of constraints on the variables of the requirement, for example: input variables where the date of birth cannot be greater than or equal to the current; height and weight cannot receive negative values; output variables where age is obtained from date of birth stored; etc. Among the artifacts produced in a Requirements Specification Document, there is the factor of complexity in analysis in: Class Diagram, identifying the attributes of classes; Data Flow Diagram, obtaining the amount of data (input, output, query, internal file and external file); Entity-Relationship Diagram, identifying the attributes of the Entities and the attributes of the Relationships; Context Diagram, through the amount of data sent or received by the external entities, among others.

2) *Number of Types of Stakeholders Involved*

As reported by Regnell, Sevensson, and Wnuk [3], one of the factors responsible for the elevation of the complexity in Requirement Engineering is the large and diversified set of stakeholders, both internal and external to the system. However, regardless of the counting of stakeholders, there is a need of classifying these types involved.

It is possible to identify in the Requirements Specification Document such factors of complexity due to: number of actors representing given types of stakeholders – possibly a wide range of stakeholders attributed to the requirement will have a greater complexity when comparing to requirements with fewer stakeholders involved, because these will be related, at least, with one system functionality, demanding to be treated to present the results intended; quantity of existing hierarchic levels for the actors – each hierarchic level created indicates the need to specify and treat the available functionalities.

Among the artifacts produces in a Requirements Specification Document, there is the factor of complexity in analysis in: Use Cases Diagram, represented by the Actors and Hierarchic Levels existing among the actors (generalization relationships).

3) *Number of External Interfaces*

The external elements, with which the software in question must interact, such as IN/OUT hardware or even other systems, are considered external resources to the software and must be treated at the requirement level. It is possible to identify the influence of this factor of complexity analyzing: number of actors representing devices, such as sensors, actuators, etc. which demand treatment to interact with the system; number of actors representing other systems; other software or systems that receive or send information to the software in question. Among the artifacts produced in requirements specification, there is the factor of complexity in analysis in the Use Cases Diagram through the identification of the Actors and Data Flow Diagram by means of external and internal entities.

4) *Functionalities Identification/Treatment*

Functionality can be defined as a behavior or an activity for which a beginning and an end can be viewed, that is, something capable of being executed. For example, the simple execution of a functionality called "perform order" refers to the activities to be performed (create order, verify customer, link product, verify stock, calculate discount, define delivery time, etc.) resulting in the creation of an instance of the entity/class called "Order". It is also recommended to present, in the description, the set of preconditions (for example, customer already registered), to implement functionality, and post-conditions (product delivered, product warranty after sale etc.) which may arise from this implementation.

It is possible to identify in the Requirements Specification Document this factor of complexity by analyzing: the number of existing functionalities to perform a requirement; necessary conditions set out in the requirement preconditions, necessary conditions set out in the requirement post-conditions, requirements that involve dependency or coupling of the functionality of other requirements. Among the artifacts produced in a Requirements Specification Document, there is the factor of complexity in analysis in: Classes Diagram, represented by the operation of classes; Data Flow Diagram, represented by the processes; Use Cases Diagram represented by the Use Cases, Requirement Specification Form, obtained from the conditions to perform a requirement; number of validations to perform a requirement, number of results obtained from the performance (main flow, requirement alternative(s) and exception (s)), among others.

D. The weights of Factors of Complexity and their Subfactors

In Table I, the factors and subfactors of the complexity proposed for MCReF are presented along with their respective weights, obtained from the results of the empirical study performed with 20 professionals from the requirements area. The factors and subfactors are objects of study and were obtained through bibliographic review of the Requirement Engineering area along with the development of case studies focused on requirements elicitation, specification and validation based on real context, among them: a) Creation of a requirement specification document, using the template Volere, referring to a monitoring and collection of a heart rate system to assess the autonomic function of the heart (in human beings); b) Creation of a requirement specification document using the templates recommended by IEEE STD 830-1998, referring to a technical and physical monitoring of athletes system on all categories of a professional soccer club [16]. To define each Weight Attributed to the Subfactors of Complexity of the Requirement, as presented in Table I, it was necessary to base on the responses obtained on the empirical study conducted with the professionals from the area. Based on the responses obtained from this study, the arithmetic average of the respondents answers were obtained for each subfactor of complexity and thus defining the subfactor Average.

TABLE I. WEIGHTS OF THE FACTORS OF COMPLEXITY OF THE REQUIREMENTS

Factor/Sub-Factor	**Sub-Factor Average**	***Weight attribute to the sub-factor of complexity of the requirements***
Q1 – Input and Output Variables		
Q1.1 – Number of Input Variables	3.60	***0.85***
Q1.2 – Number of Output Variables	3.30	***0.78***
Q1.3 – Number of Constraints of Input Variables	3.90	***0.92***
Q1.4 – Number of Constraints of Output Variables	3.60	***0.84***
Q2 – Stakeholders		
Q2.1 – Number of human actors	3.65	***0.86***
Q2.2 – Number of hierarchic levels	2.85	***0.67***
Q3 – External Interfaces		
Q3.1 – Number of actors that represent devices	3.15	***0.75***
Q3.2 – Number of actors that represent others systems	3.40	***0.80***
Q4 – Functionalities		
Q4.1 – Number of Functionalities	4.10	***0.97***
Q4.2 – Number of hierarchic levels of Functionalities	3.80	***0.90***
Q4.3 – Number of Pre-Conditions	3.55	***0.84***
Q4.4 – Number of Post-Conditions	3.40	***0.80***

To define the weight attributed to the subfactor of complexity, it was necessary to conduct, for each one, a division of the average of the subfactor obtained by the sum of the subfactors of complexity generated. With the value of the assessment of each factor of complexity, it is obtained the result, which must be multiplied by 10 (ten), to be applied in a 0-10 scale, as suggested by the metric proposed. During the empirical study, it was needed to define a weight to the factors of requirements complexity along with their subfactors of complexity, however, it was verified that only the responses attributed to the subfactors of requirements complexity would be of real interest, discharging the values obtained to the factors of requirements complexity.

The amount identified of each subfactor of requirement complexity must be multiplied by the weight attributed to the Subfactor of Complexity (SfC), resulting in the Complexity of the Subfactor of the Requirement (CSfR) and allowing them to receive their respective classification of complexity. The degree of importance of the composition to the subfactor of requirement complexity, in this study called weight of the subfactor, is the result of the empirical study conducted with the professionals of the area.

The classifications of the CSfR is the result of empirical tests conducted, and the rating value “Low” was assigned by the MCReF’s developers, based on their professional expertise; “Medium” corresponds to twice the value attributed to low classifications, “High” corresponds to higher values than the average and less than “inappropriate”. The classification “Inappropriate” indicates that the amount of elements defined for the SfC in the requirement multiplied by the weight of the factor of requirement complexity exceeds the value attributed to the value “high”. For the complexity of the subfactor of the requirement that is not identified or used in the requirement, there should be used a value of zero (0). In case there is not a CSfR classified as “Inappropriate”, it is possible to obtain the classification of the requirement by the sum of the complexities of the subfactors referring to the requirement in question, thus obtaining a “Complexity of the Requirement” (CR). This Complexity of the Requirement must be related with Table II to receive a Classification of the Complexity of the Requirement (CCR). When the CSfR is classified as “Inappropriate”, it is recommended to restructure the requirement or, “Complexity Inappropriate Requirement” must be attributed to the requirement in question, i.e., it will maintain the structure of the functional requirement in analysis, even with one or more subfactors of complexity classified as inappropriate. All Complexity of Inappropriate Requirement (CiR) indicates that one or more subfactor of complexity of the requirement was diagnosed as a number of elements defined for the SfC of the requirement that, when multiplied by the weight of the factor of requirement complexity, exceeds the value attributed to the classification “High”, then this requirement is given the Complexity Inappropriate Requirement (CiR) and its weight is the highest value shown in Table II multiplied by the number of times the SfC of requirement for the functional requirement in question was classified as inappropriate. Therefore, the **Complexity of the Requirement** is obtained by the result of the sum of the CSfR and its **Classification of the Complexity of the Requirement** is achieved through the application of the Complexity of the Requirement checked with Table II. The Classification of the Complexity of the Requirement (CCR) is the result of empirical tests grounded on the development of case studies focused on elicitation, specification and validation of requirements based on real contexts [16]. To define the classification as “Very Low” it also takes under consideration the classification “inappropriate” where both have a scale of 10 (ten) points, i.e., less than 10 points are classified as “Very Low” and the 10 points less than 100 points are “Inappropriate”.

TABLE II. CLASSIFICATION OF THE COMPLEXITY OF THE REQUIREMENT

CCR		
From	**To**	**Classification**
0	10	Very Low
11	26	Low
27	42	Middle Low
43	58	Middle
59	74	Middle High
75	90	High
91	.....	Inappropriate

III. THE EMPIRICAL STUDY THAT GROUNDED THE PROPOSED METRIC

The empirical study, which aimed the application of a questionnaire concerning to the requirements complexity identified along with the professionals of the area provided the database to obtain the weights for each factor of complexity studied. The results are shown through the following analysis: data from the participants, degree of importance attributed to the factors and subfactors of requirements complexity and reliability of the instrument of data collection.

A. *Data from the participants*

It was possible to obtain a profile of the interviewed through the part of the questionnaire “Professional Identification”. The results indicated that 100% of the participants in the empirical study were professionals with a high level of academic education, distributed in master (30%), mastering (55%) and Ph.D (15%). Regarding the time working in the area of requirements, 80% of the participants have carried out activities for 5 years or more, while only 10% has had less than a year in the area.

B. *Degree of importance attributed to the factors and subfactors of complexity of the requirement*

For the specific purpose of obtaining weights to the factors and subfactors of complexity, it was used the basic tool for data collection: a questionnaire consisting of 4 factors subdivided in 12 subfactors with 5 alternatives each, whose measures were based on Likert scale [6]. The factors considered in the empirical study were obtained by reviewing the literature about the complexity of requirements and also by the case study developed along the research using the templates Volere and IEEE STD 830-1998 to document the requirements with the factors: input variables and output of the system, Stakeholders, external interfaces to the system and system functionalities. Through this instrument to collect data, the participants were able to express their opinion about each of the affirmatives.

C. *Analysis of the Reliability*

Finished the tabulation of the research data using the statistic software SPSS (Statistical Package for the Social Sciences- version 13.0), the instrument used to collect data was subjected to a reliability evaluation through Cronbach’s Alpha coefficient analysis which works the relationship between internal covariance and variances of the measures. The value of Alfa can range between zero and one (0 - 1) and the higher this value, the greater the internal consistency of the instrument evaluated. Authors differ on the minimum acceptable value to Cronbach’s Alpha Coefficient. Hair et al. [4] said that to have an acceptable reliability, Cronbach’s Alpha must have a value of at least 0.70. However, as this is not considered an absolute value, lower values are accepted if the research is exploratory in nature. According to Malhorta [5], the minimum value of Cronbach’s Alpha to ensure the reliability in a research must be 0.60.

Using Cronbach’s Alpha in this study aimed to evaluate the internal consistency of the instrument used (questionnaire), and check if there is consistency in the variation in the participants’ responses, examining each factor and subfactor of complexity considered in the research. Table III presents the results of Cronbach’s Alpha coefficient for subfactors grouped by their factors of requirement complexity in question, i.e., involving Q1.1, Q1.2, Q1.3 and Q1.4 for Input and Output variables, Q2.1 and Q2.2 for Stakeholders, Q3.1 and Q3.2 for External Interfaces and Q4.1, Q4.2, Q4.3 and Q4.4 for functionalities.

According to the presented in this table, it is possible to observe the Alpha values obtained for each one of the factors of complexity considered in the empirical study. It is observed that the lower Alpha value produced was for the factor of Input and Output Variables (0.532) and the highest result was for the factor External Interfaces (0.834). Analyzing the general Alpha and considering all factors, it is noticed that the value generated was very satisfactory. The result indicates that the instrument used in the research is highly reliable since reached a maximum value of 1 (one), an Alpha of 0.808 was obtained. This value can be presented as an indicator of efficiency and reliability of the instrument in evaluating the factors of requirement complexity.

TABLE III. RESULTS OF CRONBACH’S ALPHA FOR FACTORS OF COMPLEXITY OF REQUIREMENT

Statistics by scale	Average	Variances	Standard Deviation	Number of variations
	3.5208	**0.9117**	**0.9548**	**4**
Cronbach’s Alpha for Factors of Complexity				
Input and Output Variables				**0.532**
Stakeholders				**0.759**
External Interfaces				**0.834**
Functionalities				**0.718**
General Cronbach’s Alpha				**0.808**

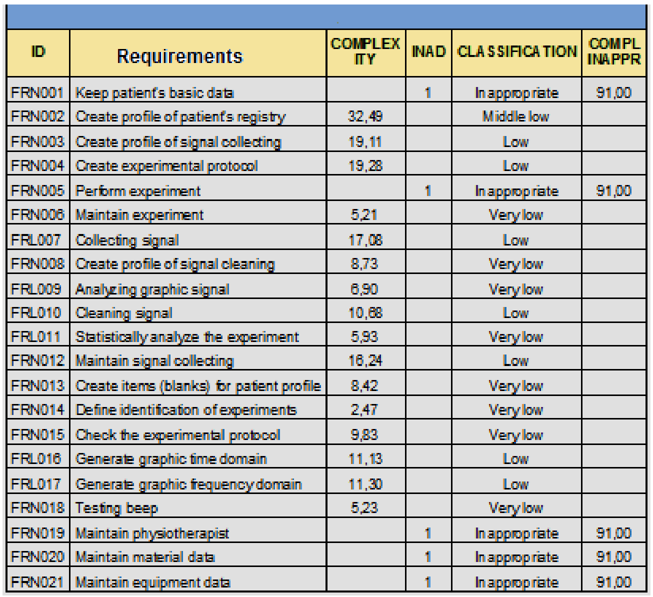
IV. CASE STUDY

The intent of this section is to present the applicability of the metrics of complexity of functional requirements – MCReF - in a case study. The context of such study was a system to monitor and capture heart rate to evaluate the autonomous function of the heart.

A. *Monitoring and Heart Rate Capturing System*

The documentation of requirements specification referring to the Monitoring and Heart Rate Capturing System to evaluate the autonomous function of the heart (in human beings) was developed by students of the Computer Science Master Degree at UNIMEP – Methodist University of Piracicaba, Brazil - related to the practical work using the Template Volere and presented to the discipline of Requirements Engineering. The documentation consists of 21 functional requirements, 15 new ones and 6 from the previous system. Table IV shows the results of the application of MCReF.

TABLE IV. RESULTS OF THE APPLICATION OF MCREF – MONITORING AND HEART RATE CAPTURING SYSTEM



ID	Requirements	COMPLEXITY	INAD	CLASSIFICATION	COMPL INAPPR
FRN001	Keep patient's basic data		1	Inappropriate	91,00
FRN002	Create profile of patient's registry	32,49		Middle low	
FRN003	Create profile of signal collecting	19,11		Low	
FRN004	Create experimental protocol	19,28		Low	
FRN005	Perform experiment		1	Inappropriate	91,00
FRN006	Maintain experiment	5,21		Very low	
FRL007	Collecting signal	17,08		Low	
FRN008	Create profile of signal cleaning	8,73		Very low	
FRL009	Analyzing graphic signal	6,90		Very low	
FRL010	Cleaning signal	10,68		Low	
FRL011	Statistically analyze the experiment	5,93		Very low	
FRN012	Maintain signal collecting	16,24		Low	
FRN013	Create items (blanks) for patient profile	8,42		Very low	
FRN014	Define identification of experiments	2,47		Very low	
FRN015	Check the experimental protocol	9,83		Very low	
FRL016	Generate graphic time domain	11,13		Low	
FRL017	Generate graphic frequency domain	11,30		Low	
FRN018	Testing beep	5,23		Very low	
FRN019	Maintain physiotherapist		1	Inappropriate	91,00
FRN020	Maintain material data		1	Inappropriate	91,00
FRN021	Maintain equipment data		1	Inappropriate	91,00

Legend:
FRN – New Functional Requirements
FRL – Legacy Functional Requirements

1) *Analysis and Discussion of the results obtained with the application of MCReF in the Monitoring and Heart Rate Capturing*

Investigating the subfactors that classify FRN001 complexity as "inappropriate", it is observed that the subfactor "number of functionalities", which presents 21 functionalities, multiplied by the weight 0.97 results to the subfactor a complexity equal to 20.37 (weight adopted according to Table I), which is higher than the stated in the classification of complexity given to the subfactor applied in the metrics, i.e., higher than 5 and less than 10.

In the analysis of the subfactors that classify the complexity of FRN002 as "middle low", it was observed that the subfactor "number of input variables" stated with 22 variables, which multiplied by the weight 0.85 results in a complexity of 18.7 to the subfactor defined as "High" in the classification of complexity.

Besides this subfactor, it was found that the subfactor "Number of Constraints to Input Variables" presents 7 variables, which multiplied by the weight 0.92 generates a complexity of 6.44 to the subfactor also defined as "High" in the classification of complexity.

Evaluating the classifications of complexity produced by the MCReF from the experience of the analyzer considering their own productivity indicator, it is observed that the result of the application of the proposed metric reflects the reality in the implementation of a software requirement, i.e., the results of the complexity obtained for the requirements corresponds to the necessary resources identified for their development and enable their identification in functional requirements of factors and subfactors of higher complexity. It is also noticed that the results obtained with the application of MCReF assist in the tasks to estimate the effort (people and professional), time and cost for development, ranging from the functional requirement of lower complexity, the FRN014, until the highest complexity, the FRN002.

V. CONCLUSION AND FUTURE WORK

With the evolution of Software Engineering techniques, it became possible to improve the software quality through standardization and definition of development processes – in accordance with the requirements – to ensure a final product that meets the customer's expectations, as agreed.

The tasks of classifying and measuring software are present from the conceptual stage (requirements) to product delivery. However, little has been explored in the Requirements Engineering area about the use of metrics of complexity. Briefly, only two studies about the subject could be identified [1][3]. Currently, there is not available, among Requirements Engineering techniques, a metric aimed specifically to measure the complexity of requirements. Such metric is of fundamental importance for software development teams in industries to have references about the degree of complexity a requirement can present. Based on a metric of complexity of requirements, the development teams can build their own productivity indicators that will be of great use to predict, with precision, variables as effort, time and cost of software development. These requirements must preferably be specified in standard documents, based on, for example, the template *Volere* or templates available in *IEEE STD 830-1998* recommendation, allowing distinguishing their main features, artifacts or diagrams contained therein, namely: treatment of functionalities; input and output; dependencies or coupling, constraints and number of stakeholders involved. With the definition of the subfactors of complexity and their respective weights and classification, it has been applied in real requirements context already specified the metric of complexity proposed. With the complexity and classification obtained for the requirements it became possible to compare the results among requirements and check the efficiency of the proposed metric. For the specific purpose of obtaining weights to the factors of complexity, it has been used a basic instrument of collecting data, a questionnaire composed of four factors of complexity, divided in 12 subfactors with 5 alternatives whose measures were based on Likert scale.

The factors considered by the empirical study were obtained through a literature review about the complexity of requirements, and also through the case studies developed along this research. Through the instrument of collecting data, the participants could express their opinion about each of the statements. The instrument used to collect data was subjected to an evaluation of reliability through Cronbach'Alpha coefficient.

Besides the evaluation of the general consistency of the instrument, Cronbach`s Alpha was employed to analyze each issue (factor and subfactor of complexity) considered in the research. Therefore, the current paper assists the development of system that use the Requirements Engineering techniques and concepts, through a metric of complexity of requirements, i.e., with the capacity of measuring how complex a requirement is, even before starting building it, identifying such complexity in the early stages of a software development life cycle. It is envisioned the possibilities of expanding this research and suggested as future works the development of a method to obtain the complexity of a set of existing requirements in a project, enabling classify the complexity of a system as a whole.

It is also suggested the development of a software to support the proposed metric. Besides such suggestions, this metric could: become a tool to estimate the cost of the software, because of the complexity involved in the requirement, being charged by the degree of difficulty for its implementation; predict the time of development of the requirement presented by the complexity associated to the resources required for implementation; estimate the delivery time of the modules of the system; establish the necessary resources (hardware, software, professionals, etc.) and qualify the software through the way of treatment of the requirement complexity. The study presented in this paper points out for the necessity of new researches in the Requirements Engineering metrics.

REFERENCES

[1] A. Kanjilal, S. Sengupta, and S. Bhattacharya, "Analysis of complexity of requirements: a metrics based approach", Proceedings ISEC'09, pp.131-132, Pune, India, 2009.

[2] Y. Zhao, H. B. K. Tan, and W. Zhang, "Software cost estimation through conceptual requirement"; Proceedings of the Third International Conference On Quality Software (QSIC'03), p.141, 2003.

[3] B. Regnell, R. Svensson, and K. Wnuk, "Can We beat the complexity of very large-scale requirements engineering?", Proceedings of the 14th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2008), pp.123-128, Montpellier, France, june/2008.

[4] J. F. Hair, Multivariate Data Analysis, 4th ed., New York: Prentice-Hall, 1995.

[5] N. K. Malhotra, Marketing Research: An Applied Orientation. New Jersey: Prentice Hall, 1996.

[6] R. Likert, "A technique for the measurement of attitudes". Archives of Psychology, 1932.

[7] B. W. Boehm and P. N. Papaccio, "Understanding and controlling software costs", IEEE Transactions on Software Engineering, p.1462-1477, vol. 14, no. 10, 1988.

[8] N. E. Fenton and S. L. Pfleeger, Software Metrics – A Rigorous and Practical Approach, 2nd ed., PWS Publishing Company, 1997.

[9] IEEE, IEEE Std 830-1998 Software Requirements Specification, The Institute of Electrical and Electronics Engineers, New York, 1998.

[10] S. H. Kan, Metrics and Models in Software Quality Engineering; Addison–Wesley, 2002.

[11] G. Kotonya and I. Sommerville, Requirements Engineering: Processes and Techniques; John Wiley & Son, Chichester, England, 1998.

[12] S. Park and J. Nag, "Requirements management in large software system development"; IEEE International Conference on Systems, Man and Cybernetics, pp. 2680-2685, vol. 3, 1998.

[13] R. S. Pressman, Software Engineering: A Practitioner´s Approach; McGraw-Hill, 7th edition, 2009.

[14] J. Robertson and S. Robertson, Volere: Requirements Specification Template, Edition 14, 2009.

[15] I. Sommerville, Software Engineering; Addison Wesley, 9th edition, 2010.

[16] C. R. Paviotti and L.E.G. Martins, MCReF – Métrica de Complexidade de Requisitos. Revista Conteúdo, vol.1, no.6, pp.1-26, ago/dez 2011.

Hierarchical Multi-Views Software Architecture

Ahmad Kheir, Mourad Oussalah
LINA Laboratory
Nantes University
Nantes, France
{Ahmad.Elkheir, Mourad.Oussalah}@univ-nantes.fr

Hala Naja
LaMA Laboratory
Azm Center For Research, Lebanese University
Tripoli, Lebanon
Hala.Naja@ul.edu.lb

***Abstract*—Software design and development hold so many inconsistencies when it comes to build composable and scalable structures. However, software architectures could be an efficient solution if considered with additional features like the composition of such architectures by linking different hierarchized views formally together. Thus, this paper presents a new contribution of a multi-views/multi-hierarchy software architecture that is consistent with the ISO/IEC/IEEE 42010 standard, and that presents a way for defining formally the consistencies between its different views and hierarchy levels.**

Keywords-Software architecture; Views; Hierarchy levels; Consistency

I. INTRODUCTION

Software architectures have contributed effectively in complex and distributed software systems development. Normally, there are two principles, which have made the software architectures' contribution obvious and indispensable. First, it allows the architect to model the structure and the behavior of the system simultaneously. Second, it offers the architect the base to build multi-hierarchy based models.

In fact, coherent and well organized software architecture would enhance some crucial system properties like the reliability, consistency, and scalability. However, the lack of such architectures may limit those systems' adaptability, evolution, and consequently their life cycle, due to the incapability of modifying or expanding the stakeholders' requirements.

This paper presents a Model, View and Abstraction Level based software architecture (MoVAL), a multi-views and multi-hierarchy software architecture, which complies with the IEEE standard 42010-2011 [1] and is based on the construction of multi-views models having for each of their views a hierarchy of levels.

Actually, the concept of viewpoint was present in many fields of software engineering domain. Indeed, it was introduced in requirements engineering by A. Finkelstein [4] in 1989 opening the way for other valuable works in this field like in [5] and in [6]. Also, the viewpoint concept was existing in software modeling, implicitly in some cases like in the unified modeling language (UML), where each diagram type has an implicit viewpoint, and explicitly in other studies like in the View-based UML extension (VUML) [7], where an explicit representation of different viewpoints in a single multi-views class diagram is proposed. Also, the software implementation field recognized the utility of viewpoint concept. Indeed, different development paradigms encapsulate the viewpoint concept, like the aspect oriented [8], subject oriented development paradigms [9] and the view-based programming technique [10], which define explicitly different views in a single model. In addition, most of the related works done in the field of software architecture like the *4+1 View Model* [2] and the *Views and Beyond* [3] approaches, have defined multi-views software architecture. However they did not provided any type of hierarchy for their views in order to reduce their complexities, nor they defined formally some consistency rules between different views of an architecture in order to conserve the robustness of that architecture and its ability to evolve while the stakeholders' requirements evolve. A complete survey on related works and a fruitful analysis of their limitations was presented in a previous study [11], but we can summarize those limitations in three main points: the views inconsistencies, the need to move between different abstraction levels, and the lack of a complete architectural description process.

In light of the related works study, MoVAL's motivations and goals were made clear. Actually, there are two main goals that were intended in this approach. The first goal is to propose a multi-views software architecture defining for each view a multi-levels hierarchy aiming to minimize software systems complexity per modeling entity. The second goal addressed in this approach, is to define formally the relationships that may exist between different views of a model, and also between different hierarchy levels inside a given view.

This paper is organized as follows: Section II presents in details our contribution. Then, the proposed approach is illustrated by a case study in Section III. Finally, Section IV concludes the paper.

II. MOVAL

In MoVAL, a model is conceptualized via a matrix as illustrated in Figure 1.

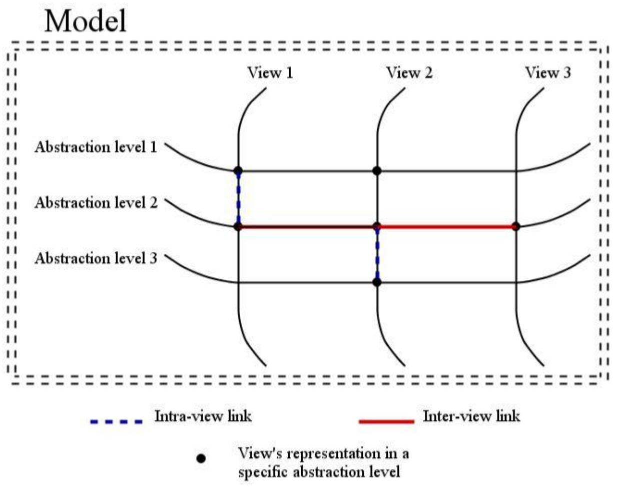


Figure 1. Conceptual Matrix of a MoVAL model.

The columns of the matrix represent the views of the model, while the lines represent its abstraction levels, which are the first level of the views' hierarchy detailed further in this paper. Hence, the lines and columns of the matrix illustrate two distinct structuring types defined in MoVAL. The columns illustrate the vertical structuring referring to different views of the same model, and the lines illustrate the horizontal structuring referring to the hierarchy levels defined in the model and associated to its views.

Note that model's matrix, in some trivial cases where the architect decides to create only one view for the model, and decide to represent this unique view in a single abstraction level, could be reduced to a single element.

A. *Model View*

A model view in MoVAL, or simply a view, is a representation of this model considering, from one side, a set of the development process' aspects, and from another side certain problems associated to a specific category of stakeholders or a group of categories of stakeholders. Those development aspects and problems are grouped in a separate entity, named viewpoints. In general, every stakeholder needs to express his interests via some appropriate semantics, syntax, and tools, called formalisms. For example, a database administrator needs to use the entity-relationship diagrams (ERDs) and the appropriate tools in order to model his database in a given phase. Thus, a viewpoint also defines the formalisms that shall be used afterwards to model the inherent views. Hence, each view must be associated to a specific viewpoint, which should be either predefined like the physical, structural, and behavioral viewpoints, or customized based on the application domain like the thermic view in an automobile construction system.

B. *View's Hiearachy Level*

MoVAL approach has defined a hierarchy of levels for each view, in order to describe it formally and appropriately in each step of the development process.

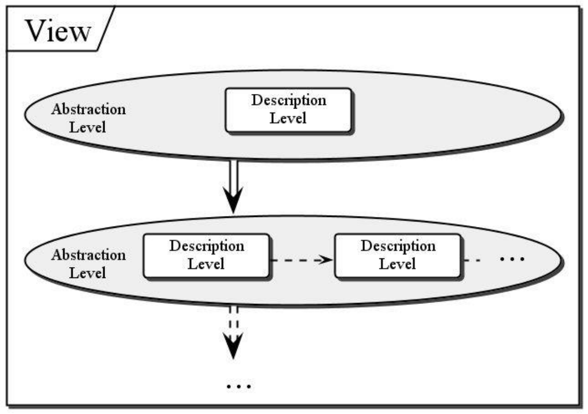


Figure 2. Views and hierarchy levels.

Figure 2 represents this hierarchy, which consists of two types of levels, the abstraction levels, which are represented in the figure via ovals. Also, under each abstraction level several description levels are represented via correlated rectangles.

1) *Abstraction Level*

An abstraction level is a representation of a view considered at a specific stage of the system lifecycle. Eventually, several abstraction levels could be considered on the same view, and then linked together by higher/lower relationships. In fact, for the same view, an abstraction level AL1 is higher than another abstraction level AL2 (*resp.* AL2 is lower than AL1) if AL1 defines relevant requirements in a given stage of the system lifecycle leaving out some other requirements and relegating them to AL2 in a more advanced stage.

For a given view, an abstraction level must use appropriate formalisms that are implied by the associated viewpoint.

In general, a view could have more than one abstraction level having the same inherent requirements as long as they have different formalisms. Actually, in this case the transition from one abstraction level of a view to another abstraction level in the same view conserving the same inherent requirements and changing the formalism, could indicate the transition from a stage of the software lifecycle to another more advanced stage.

Note that it is not mandatory to have always an isomorphism between different views of a model, by the fact that it is not mandatory to have each abstraction level associated to all the views of the model, as illustrated in Figure 1.

2) *Description Level*

The second type of hierarchy levels of a view is the description level. This type of hierarchy levels allows the architect to describe the same abstraction level of a specific view and the same inherent requirements while providing multiple descriptions having different granularity levels.

Here also, the description levels of the same abstraction level are linked together by higher/lower relationships. So, a description level DL1 is higher than another description level DL2 (*resp.* DL2 is lower than DL1) if DL1 lies on the same requirements as DL2 but adds more details in order to make easier the understanding of DL2's requirements. In other words, DL1 is at a higher granularity level than DL2.

Actually, the difference between this type of hierarchy levels and the abstraction levels, resides in the fact that a lower abstraction level allows the architect to go straightforward into more advanced stages of the system lifecycle relatively to the higher abstraction level, in general, by providing more requirements. However, a lower description level does not allow the architect to provide additional requirements of a specific view, but it allows him to describe more clearly its previous description level by providing more description details.

C. Link

The links are structural elements defined in MoVAL in order to express formally the relations between different hierarchy levels and conserve model's consistency. Those links are grouped in four categories:

- **Inter-views link**, defining the relation among a couple of distinct hierarchy levels belonging to two different views.
- **Inter-levels link**, defining a similar relation to that defined by the inter-views link, except that the hierarchy levels here belong to the same view.
- **Intra-level link**, defining an internal relation between elements of the same hierarchy level.
- **User links**; this category of links is a special category. A user link always inherits from one of the three previous categories, then defines some additional structural or semantic properties and attributes (see the case study in Section IV). Actually, the purpose of this category of links was to enhance the modularity and reusability of software architecture's structural elements.

In order to formalize the links, MoVAL has attributed four main properties to define them:

- **Source**: based on the semantic role of a link, its source could be either an abstraction or a description level of a view.
- **Destination**: similarly, the type of the destination of a link depends on its semantic role. Note that always the source and destination of a link must have the same type.
- **Semantic role**: the semantic role of a link defines the nature or the purpose behind the relation between the source and destination hierarchy levels. It is firmly related to the category of the link and the type of its source and destination hierarchy levels. Hence, MoVAL has defined three main semantic roles:
    - **Connection**, specifying some consistency rules between elements of the same hierarchy levels. Note that this semantic role could be used only for intra-level links.
    - **Composition**, specifying the composition of elements of the source level in the destination level, which is in this case the lower level. This role could be used in case of inter-levels or inter-views links.
    - **Expansion**, representing the description of elements of the source level in the destination level, which is in this case the lower level, respecting the abstraction levels of the source and destination. Actually, this semantic role is dedicated for the representation of relations between abstraction levels only and could be used in both cases of inter-levels or inter-views links.

  Normally, composition and expansion roles are adequate when the architect adopts a Top-Down development strategy. However, when the Bottom-Up strategy is adopted, composition and expansion could be replaced by other roles having inverse semantics, which are respectively the aggregation and compression semantic roles.
- Semantic link, which includes a set of semantic attributes aiming to implement the desired semantics, chosen in advance by architect via the semantic role:
    - **Dependence**, declaring that the destination hierarchy level depends for its existence on the source hierarchy level.
    - **Predominance**, which declares semantics symmetric to those declared by the dependence attribute.
    - **Coherence**, specifying that some consistency rules should be considered and respected in the destination hierarchy level based on the source level parameters, in order to conserve the coherence of the model. Those consistency rules could be expressed via a given constraint language like OCL.

D. MoVAL Meta-Model

MoVAL meta-model is consistent with the ISO/IEC/IEEE 42010 standard. Thus, some elements have kept their definitions presented in the IEEE standard, like the definition of a system, architecture, architectural description, stakeholder, viewpoint, view, and concern. However, some other elements were given new definitions like the model, and others have been introduced like the abstraction and description level, formalism, and link. Figure 3 presents the proposed meta-model.

A *System*, as it was defined in the IEEE standard, is not limited to individual applications but it encompasses them to cover also the subsystems, systems of systems and all kind of software interests' aggregations. A system always has different categories of *Stakeholders*, which are the participants in every phase of his life cycle. They could be individuals, teams or even organizations interested in this system, like the system architects, developers, analysts, experts contributing in the system development, users, etc.

Each of those stakeholders focuses on a specific part of the system requirements saturating his interests. Hence, those interests of different stakeholders are defined as different sets of *Concerns* overlapping in certain cases and contradicting in other cases.

Simultaneously, a system is associated to an *Architecture*, documented and described via an *Architectural*

Description (AD). An AD is composed of a set of *Views* governed by a set of *Viewpoints* specifying and grouping the inherent concerns and formalisms that should be used for the development of the views. Those views are represented in a hierarchy of *Abstraction* and *Description Levels*.

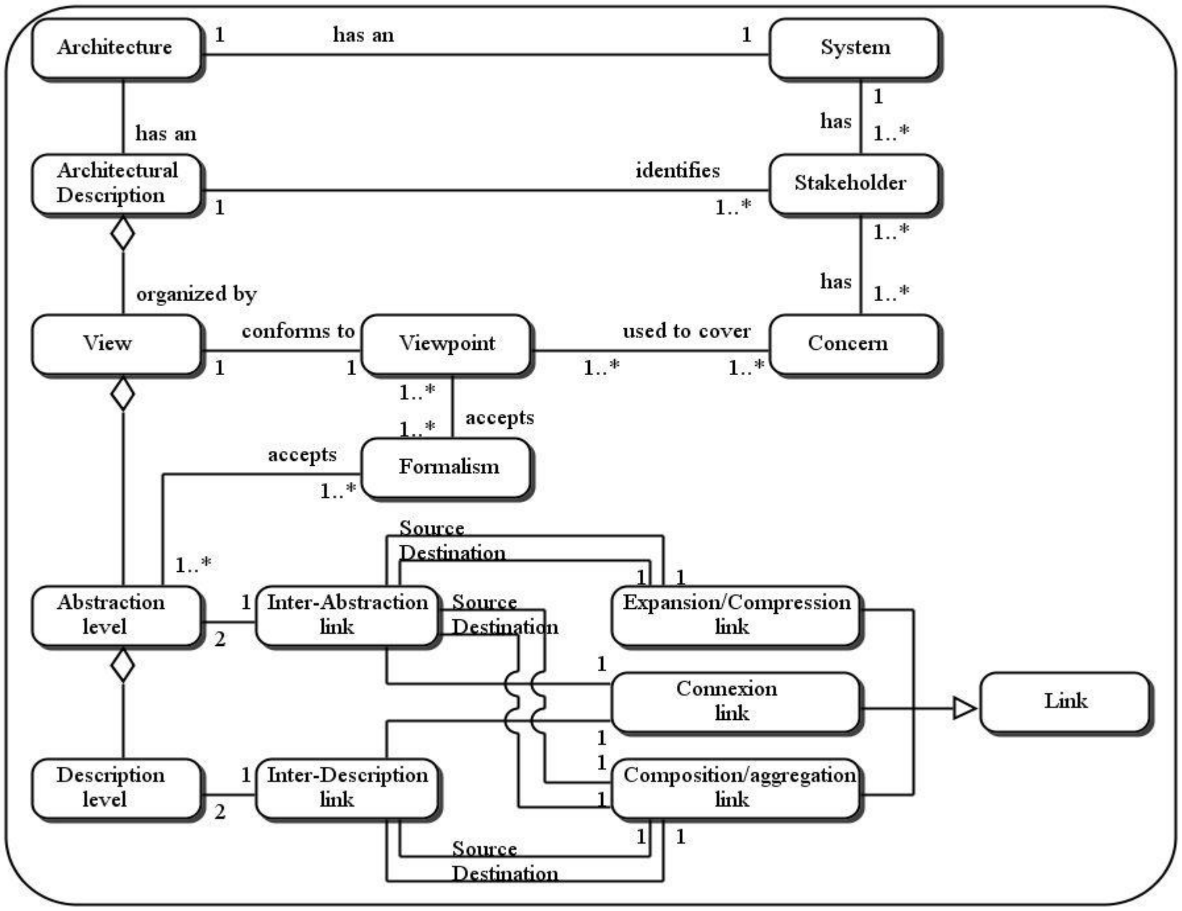


Figure 3. Conceptual model of MoVAL

Also, each viewpoint and each abstraction level of a model offers a set of Formalisms that could be used afterward to model the associated view at each of its abstraction levels. Those formalisms define actually the lexical and syntax elements that could be used.

III. Case Study

In order to clarify MoVAL concepts and confirm its contribution and utility in software engineering and complex systems development field, a case study will be represented in this section.

This case study consists on an eCommerce WebApp, in which multiple stores would be registered and given virtual spaces to expose their products for sale.

In this context, only three viewpoints are considered (due to space limitation issue):

- ✓ **Physical** viewpoint, which represents the view of the system deployer. Thus, it manipulates the hardware and software resources used for the deployment of such systems. Actually, this viewpoint is predefined in MoVAL and considered associated to a single formalism, which is the deployment diagram of UML. The associated view could be represented in a hierarchy of one abstraction level and one description level mentioned respectively in figures 4 and 5.

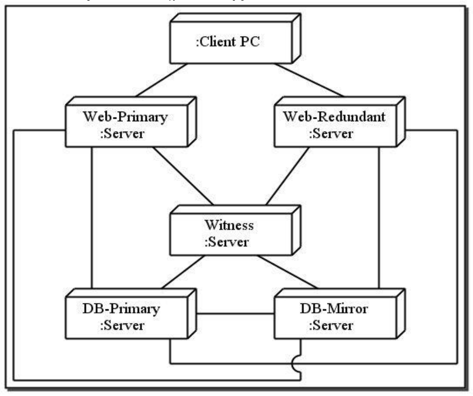


Figure 4. Physical view abstraction level.

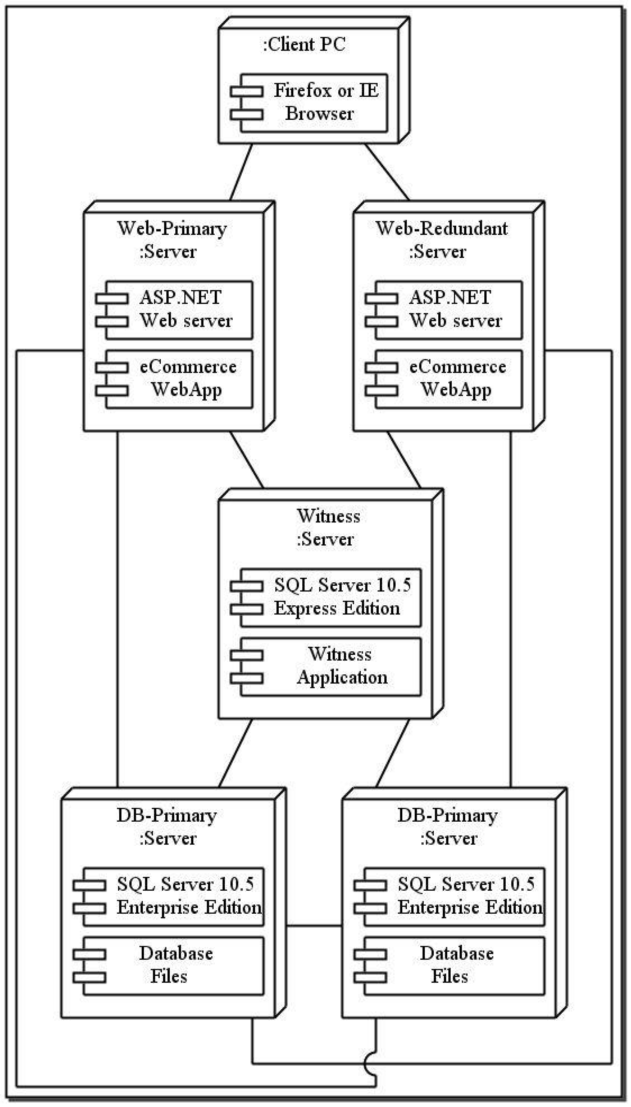


Figure 5. Physical view description level

- ✓ Site administrator viewpoint, representing the system as seen by the system administrator and considering his requirements. Three formalisms could be associated to this viewpoint, which are the use case, sequence, and class diagrams of UML. In addition, the associated view could be defined in two abstraction levels illustrated in figures 6 and 7, respectively.

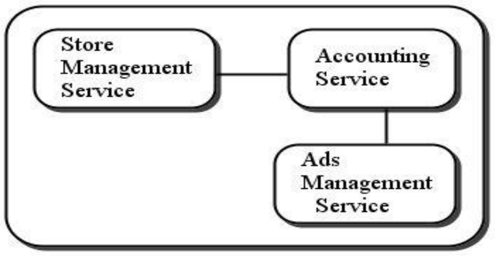


Figure 6. First abstraction level of the Site admin view.

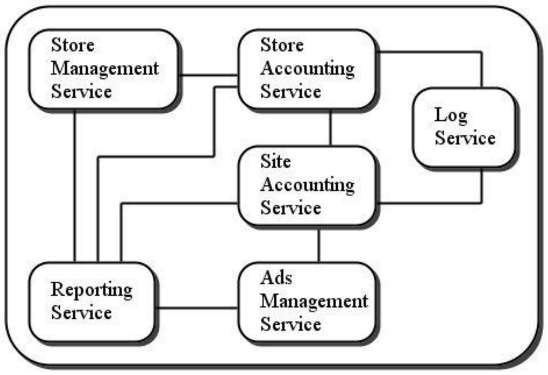


Figure 7. Second abstraction level of the Site admin view.

- ✓ Store administrator viewpoint, representing the system as seen by the registered store administrator. This viewpoint will be associated to the same formalisms associated to the previous viewpoint, also the associated view will be defined in two abstraction levels illustrated in figures 8 and 9.

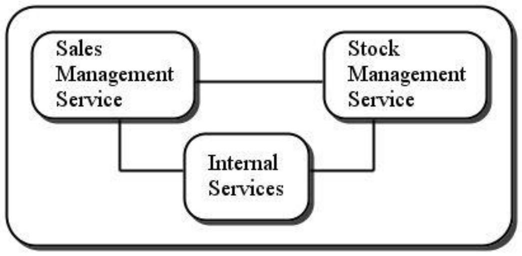


Figure 8. First abstraction level of the Store admin view.

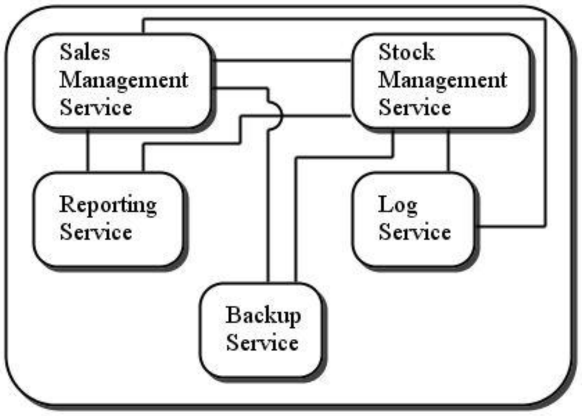


Figure 9. Second abstraction level of the Store admin view.

In general, in order to improve the models' consistency, the system architect must create different links between different views and hierarchy levels of this model. For this reason, the abstraction levels of the *Site Administrator* view could be associated to the abstraction levels of the *Store*

Administrator view, as they share the same level of details. Thus, for the remaining of this section, the higher and lower abstraction levels, associated to this couple of views, will be referred by the *First Functional Level* and the *Second Functional Level*, respectively.

Now, three links, among others, could be derived for this case study:

- ✓ Inter-levels link having the *First Functional Level* of the *Site Administrator* view as source hierarchy level, and the *Second Functional Level* of the same view as destination. This link is a composition link expressing in his coherence semantic attribute the composition of the *Accounting Service* in the source by the *Site Accounting Service* and the *Store Accounting Service* in the destination.
- ✓ Inter-levels link having the *First Functional Level* of the *Site Administrator* view as source and the *Second Functional Level* of the same view as destination. This link is an expansion link expressing in his coherence semantic attribute the expansion of the *Internal Services* of the source level to the *Log Service* and the *Backup Service* in the destination.
- ✓ User link, named *Reuse Link*, created by the architect as an inter-views link defining the reusability of a component of the source level in the destination level. Hence, a Reuse link could be defined having the *Second Functional Level* of the *Site Administrator* view as source and the *Second Functional Level* of the *Store Administrator* view as destination. This link expresses the reusability of the *Reporting Service* in both of the source and destination levels.

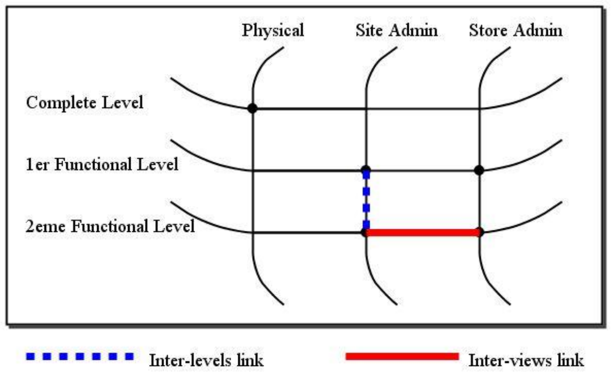


Figure 10. Conceptual matrix of the eCommerce model.

Figure 10 represents the conceptual matrix of the eCommerce case study.

IV. CONCLUSION

This paper has presented a new contribution of multi-views and multi-hierarchy software architecture, named MoVAL, defining and modeling independently, for each stakeholder, its inherent concerns in a separate multi-levels view and providing the necessary definitions to combine and link all those views and hierarchy levels in order to guaranty a complete consistency between different parts of the resulting architecture.

In fact, MoVAL has given every stakeholder the space to model his interests and the tools to represent the possible interferences that may exist with other interests of other stakeholders, what should decrease significantly the number of unexpected executions or the number of bugs of the system, and increase consequently the system's reliability.

From another side, MoVAL has given the software architect the tools to link different semantically related views or abstraction levels via the architectural links, what would enhance the model coherence because of the representation of every constraint that may exist between different views or abstraction levels. Simultaneously, this organization and coherence make the addition of other user requirements much simpler, and consequently increase model's scalability.

Actually, MoVAL is in the prototyping phase. A specific framework encapsulating the all the tools and features needed to apply MoVAL's concepts will be implemented and validated.

ACKNOWLEDGMENT

This material is based upon work supported by the Lebanese association for scientific research (LASeR), the Lebanese council for scientific research (CNRS), the French association CAPACITES, the University of Nantes, and finally the Lebanese University.

REFERENCES

[1] ISO/IEC/IEEE, "Systems and software engineering -- Architecture description," in ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), 2011.

[2] P. B. Kruchten, "The 4+ 1 view model of architecture," IEEE Software, vol. 12, no. 6, 1995, pp. 42–50.

[3] P. C. Clements et al., "A practical method for documenting software architectures," Research showcase, Carnegie Mellon University, 2002.

[4] A. Finkelstein and H. Fuks, "Multiparty specification," in ACM SIGSOFT Software Engineering Notes, vol. 14, 1989, pp. 185–195.

[5] B. Nuseibeh, J. Kramer, and A. Finkelstein, "A framework for expressing the relationships between multiple views in requirements specification," IEEE Transactions on Software Engineering, vol. 20, no. 10, 1994, pp. 760–773.

[6] I. Sommerville and P. Sawyer, "Viewpoints: principles, problems and a practical approach to requirements engineering," Annals of Software Engineering, vol. 3, no. 1, 1997, pp. 101–130.

[7] M. Nassar et al., "VUML: a Viewpoint oriented UML Extension," Proceedings of the 18th IEEE International Conference on Automated Software Engineering (ASE'03), IEEE Computer Society, 2003, pp. 373–376.

[8] D. Majumdar and S. Bhattacharya, "Aspect Oriented Requirements Engineering: A Theme Based Vector-Orientation Model," Infocomp J. Comput. Sci., vol. 9, no. 1, 2010, pp. 61–69.

[9] H. Ossher, M. Kaplan, W. Harrison, A. Katz, and V. Kruskal, "Subject-oriented composition rules," in ACM SIGPLAN Notices, vol. 30, no. 10, 1995, pp. 235–250.

[10] H. Mili, J. Dargham, A. Mili, O. Cherkaoui, and R. Godin, "View programming for decentralized development of OO programs", in Technology of Object-Oriented Languages and Systems (TOOLS 30), 1999, pp. 210–221.

[11] A. Kheir, H. Naja, M. Oussalah, and K. Tout, "Overview of an Approach Describing Multi-Views/Multi-Abstraction Levels Software Architecture," Proceedings of the 8th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2013), France, 2013, pp. 132–140.

Object Oriented Petri Nets in Software Development and Deployment

Radek Kočí and Vladimír Janoušek

Brno University of Technology, Faculty of Information Technology,
IT4Innovations Centre of Excellence
Bozetechova 2, 612 66 Brno, Czech Republic
{koci,janousek}@fit.vutbr.cz

***Abstract*—Modeling, implementation, and testing are integral parts of system development process. Models usually serve for description of system architecture and behavior and are automatically or manually transformed into executable models or code in a programming language. Tests can be performed on implemented code or executable models; it depends on used design methodology. Although models can be transformed, the designer has to usually adapt resulted code manually. It can results in inconsistency among design models and their realization and the further development, testing and debugging by means of prime models is impossible. The approach discussed in this paper allows to model and test systems using high-level languages, especially Object Oriented Petri Nets combined with Discrete Event System Specification, whereas models are deployed to the product environment and become integral part of the system.**

Keywords-Object Oriented Petri Nets; DEVS; model deployment.

I. Introduction

Modeling, implementation, and testing are integral parts of system development process. Various models are used in analysis and design phases and usually serve as a system documentation rather than real models of the system under development. The system is then implemented according to these models, whereas the code is either generated from models or is implemented manually. Unfortunately, many implementation differ from designed models because of debugging or system improvement. Consequently, models become out of date and useless.

To solve a problem with manual implementation and impossibility to test designed system using models, the methodologies and approaches commonly known as Model-Driven Software Development are investigated and developed for many years [1], [2] These methods use executable models, e.g., Executable UML [3] in Model Driven Architecture methodology [4], which allows to test systems using models. Models are transformed into another models and, finally, to code. Nevertheless, the resulted code has to often be finalized manually and the problem with semantic mistakes or imprecision between models and transformed code remains unchanged.

The approach to system development, which is presented in the paper, uses formal models as a means for system description as well as system implementation. The basic idea is to have a framework allowing to execute models in different modes, whereas each mode is advisable for another kind of usage—design, testing, and deployment. The system is developed using different kinds of models (from formal models to direct code in a programming language) in simulation, i.e., it is possible to test systems in any state in any time. The design method, which is taken into account in the papers [5][6], does not require model transformations and assumes that models serve for system description as well as system implementation. The formalism of Object-Oriented Petri Nets (OOPN) [7], [8] and Discrete Event System Specification (DEVS) are basic modeling means.

The paper is organized as follows. First, we briefly introduce the used formalisms of OOPN and DEVS in Section III, application framework in Section IV, and design methodology including a simple case study model in Section V. Possibilities to deploy models into product environment will be discussed in Section VI.

II. Related Work

Combination of formal models, simulation, and model deployment is applicable mainly in control software. The use of high-level languages, especially Petri Nets, allows to build and maintain control systems in a quite fast and intuitive way. To control robot application, hierarchical binary Petri nets are used for middleware implementation in a RoboGraph framework [9]. To develop control software for embedded systems, the work which uses Timed Petri Nets for the synthesis of control software by generating C-code [10], the work based on Sequential Function Charts [11], or the work based on the formalism of nets-within-nets (NwN) [12], [13], [14] can be mentioned.

These tools and works allow to *model* systems using a combination of different formalisms, but do not allow to use formal models in system *implementation*. The proposed approach allows to use formal models as a basic design, analysis and programming means combining simulated and real components. The main advantages; there is no need for code generation, and for further investigation of deployed systems, using the same formal models and methods is possible.

III. Used Formalisms

We will briefly introduce the formalisms of Object-Oriented Petri Nets and Discrete Event System Specification in this section.

A. Formalism of Object Oriented Petri Nets

Object orientation of Object-Oriented Petri nets (OOPN) [15] is based on the well-known class-based approach. All objects are instances of classes, every computation is realized by message sending, and variables contain references to objects. This kind of object-orientation is enriched by concurrency. OOPN objects offer reentrant services to other objects and, at the same time, they can perform their own independent activities. The services provided by the objects as well as the autonomous activities of the objects are described by means of high-level Petri nets—services by *method nets*, object activities by *object nets*.

The formalism of OOPN contains important elements allowing for testing object state (*predicates*) and manipulation with object state with no need to instantiate nets (*synchronous ports*). Object state testing can be negative (*negative predicates*) or positive (*synchronous ports*). We can see that synchronous ports can be used for testing as well as for manipulation. *Synchronous ports* are special (virtual) transitions, which cannot fire alone but only dynamically fused to some other transitions, which activate them from their guards via message sending. *Negative predicates* are special variants of synchronous ports with inverted semantics—the calling transition is fireable if the negative predicate is not fireable.

B. Formalism of DEVS

Discrete Event System Specification (DEVS) [16] is a formalism, which can represent any system whose input/output behavior can be described as sequence of events. The atomic DEVS model is specified as a structure M containing sets of states S, input and output event values X and Y, internal transition function δ_{int}, external transition function δ_{ext}, output function λ, and time advance function ta. These functions describe behavior of the component.

This way we can describe atomic models. Atomic models can be coupled together to form a coupled model CM. The later model can itself be employed as a component of a larger model. This way the DEVS formalism brings a hierarchical component architecture. Sets S, X, Y are obviously specified as structured sets. It allows to use multiple variables for specification of a state; we can use a concept of input and output ports for input and output events specification, as well as for coupling specification. In another words, components are connected by means of ports and event values are carried via these ports.

IV. Application Framework

Since one of the main motivations behind the development of OOPN is a possibility to use Petri nets not only for system modeling but also for system implementation and deployment, we need an application framework, which fulfils two basic requirements. First, to link models and product environment. Second, to work with models in simulations.

A. Interoperability with Product Environment

The models described by means of OOPN can cooperate with objects of the product environment (product objects). Since the framework is implemented in Smalltalk [17], OOPN objects can send messages to Smalltalk objects, and OOPN objects can be directly available in Smalltalk. There are different levels at which the product objects can send messages to OOPN objects—*domain*, *predicate*, and *synchronous port* levels. Domain level allows Smalltalk objects to send messages OOPN objects as though they were Smalltalk objects. Predicate level allows to test predicates and port level allows to perform synchronous ports. Each OOPN object offers special meta-protocol allowing to work at presented levels (it will be shown in the text, later on).

Another way on how to connect OOPN models with their product environment is to use component approach based on DEVS formalism. DEVS component can wrap another kind of formalism, so that each such a formalism is interpreted by its simulator and simulators communicate each other by means of a compatible interface. Let $M_{PN} = (M, \Pi, map_{inp}, map_{out})$ be a DEVS component M, which wraps an OOPN model Π. The model Π defines an initial class c_0, which is instantiated immediately the component M_{PN} is created. Functions map_{inp} and map_{out} map ports and places of the object net of the initial class c_0. The mapped places then serve as input or output ports of the component.

B. System in Simulation

The framework offers a protocol for creating and manipulating models and simulations. Models are usually described by formalisms of OOPN or DEVS, but can be implemented in product environment or can interoperate with product environment. The framework allows to execute models in different simulation modes—simulation in *model time*, simulation in *real time*, and simulation in *combined time*.

Each simulation mode is advisable for another kind of usage. *Model time* is intended for basic design, testing, and analysis of system under development and assumes all components are described by formal models. *Combined time* assumes that the system is descibed by formal models as well as implemented in product environment, i.e., selected simulated components are replaced by their real implementation, whereas simulated components work in model time and real components work in real time. This mode allows to experiment with simulation models in real conditions. *Real* *time* assumes that all components (simulated as well as real) work in real time and is intended for hardware/software-in-the-loop simulation and system deployment.

V. System Modeling Using OOPN and DEVS

The system is modeled and simulated in the application framework, which supports formalisms of OOPN and DEVS, so far. This section will demonstrate modeling methodology based on usage of the application framework.

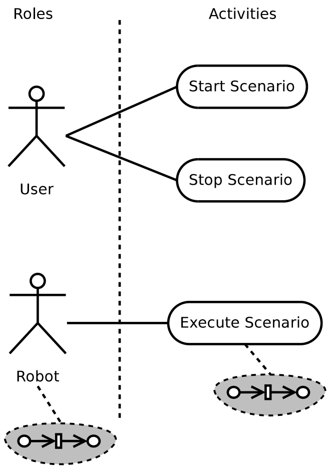


Figure 1. Use Cases of designed system.

A. Modeling Methodology

We will follow the design methodology, which has been presented by Kočí and Janoušek [18]. The modeling process starts with identification of *actors* and *use cases* as a model of *system behavior*. In this phase, the use case diagrams from UML can be used. Based on this diagram, *roles* and their *activity nets* are defined. Roles are based on analysis of actors (actors usually correspond to roles) and activity nets model behavior described by use cases.

Next step is to define an architecture of the system. The architecture can be described by class diagram. Roles and activity nets are encapsulated into classes, furthermore the *subjects* are identified and modeled using classes. Subjects represent information about actors or a group of actors, e.g., one user (a subject) can have more roles (administrator, customer, etc.). The architecture is based on layered modeling of roles and their activities, i.e., each activity encapsulates a role, an activity can encapsulates another activity, etc. Each role and its set of allowed activities (activity nets) can be described by any formalism allowing to define an interface for communication or synchronization, e.g., statecharts, activity diagrams, Petri Nets, etc.

B. System Behavior Modeling

We will demonstrate system modeling and model deployment on a simple case study of a robot control system. First, we identify use cases of the system, as shown in Figure 1. We have found two actors (*User* who can control the system and *Robot* who is controlled) and three use cases (*Execute Scenario* for *Robot*, and *Start Scenario* and *Stop Scenario* for *User*).

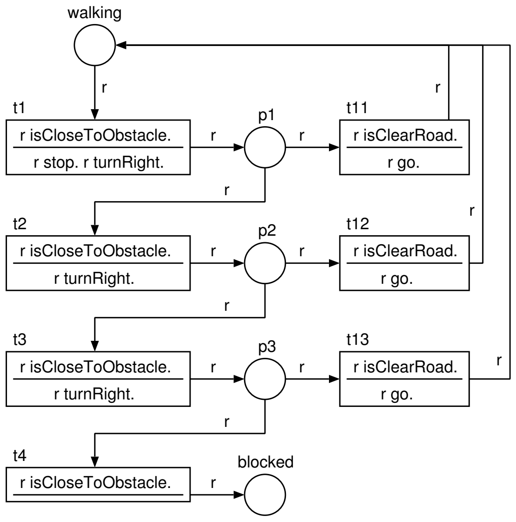


Figure 2. Activity Net *Scenario*.

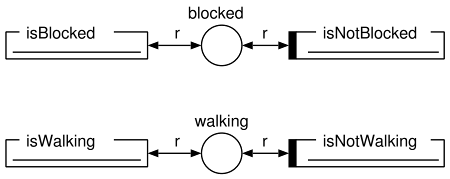


Figure 3. Activity Net *Scenario* – predicates.

Actors represent *roles* and use cases represent *activities* in the system. We aim at the actor *Robot* and the use case *Execute Scenario* in our study. The use case models an activity of the robot. We will suppose very simple activity, which can be described in following algorithm: (1) the robot is walking, (2) if the robot comes upon to an obstacle, it stops, turns to right and tries to walk, (3) if the robot turns three times with no possibility to walk, it stops. The activity net *Scenario* describing the presented behavior of use case *Execute Scenario* is shown in Figure 2.

The robot can be in two stable states—walking or blocked (there is no possibility to walk). Each such a state is represented by appropriate place, i.e., places `walking` and `blocked`. We have to be able to test activity states, therefore the predicates are generated for each such a place—the synchronous port `isBlocked` and the negative predicate `isNotBlocked` for the state `blocked` and similar predicates for the state `walking`. Test predicates are shown in Figure 3.

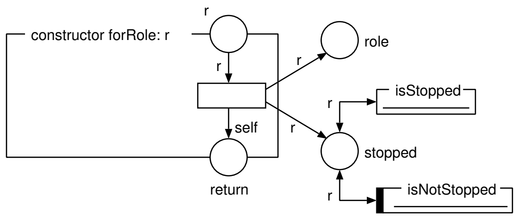


Figure 4. Activity Net *Scenario* – the constructor.

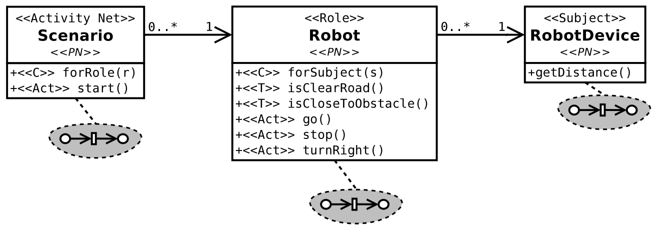


Figure 6. Basic architecture of the case study.

The activity has to be linked to a role in the system—this role is stored in places and serves even as a state token. The role supplies an information about the robot and allows to send commands. Each activity is instantiated for just one role, so that the role is initialized by means of constructor as shown in Figure 4. The new state `stopped` is added—it represents a situation when the robot is stopped but does not stay before any obstacle (e.g., the robot was stopped by user or the activity is being created).

Now, we have to add last element, a possibility to start activity—it is a part of use case *Start Scenario* modeled by method net `start` (see Figure 5), which decides what has to be done based on the activity state. If the activity is *walking*, the method does nothing. If the activity is stopped or blocked, it starts the robot's walk (send a message `go` and moves the token to the place `walking`.

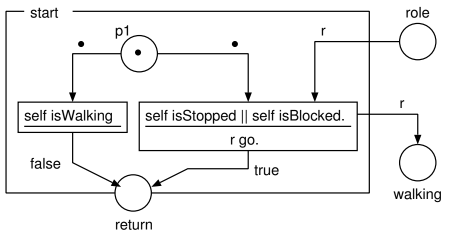


Figure 5. Activity Net *Scenario* – a method net *start*.

C. Architecture Modeling

Each role needs to have its subject, i.e., the object defining information about a subject, which can have different roles in the system. The subject is usually modeled as an object containing efficient data directly or as an interface to database, another system or remote object. The way how to model subjects influences the system architecture.

Figure 6 shows the classes of basic architecture of our example with appropriate stereotypes *Activity Net*, *Role*, and *Subject*. The architecture consists of the subject *RobotDevice*, its role *Robot* and its activity *Scenario*, that have been modeled by OOPN (see the stereotype *PN*). *RobotDevice* represents an interface to the simulated robot and *Robot* represents a role which the robot has in the system. Each method is labeled with one of stereotypes `C` (constructor), `Act` (activity), and `T` (testing) determining a realization of methods in OOPN (see [19]).

D. DEVS Architecture Modeling

The DEVS architecture of presented case study contains the components *Behavior* and *Subject* as shown in Figure 7. The component *Behavior* describes the system behavior as presented in previous case and the component *Subject* describes a subject of behavior. Subcomponents of the component *Subject* can be modeled by OOPN, programming language, or any other supported formalism. Components are connected via ports *request* and *answer*. The DEVS subcomponent *RobotDevice* is an atomic component, which gets a request string at its input port $request$, asks a robot for answer, and puts this answer to its output port $answer$. This architecture allows to exchange components in a very simple way, because components are connected only by means of ports.

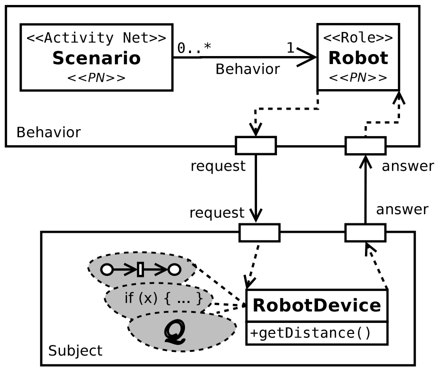


Figure 7. DEVS architecture of the case study.

VI. Software Deployment with Models

This section will demonstrate possibilities of keeping models in the deployed system. It is based on the application framework allowing to interoperability of models and product environment.

A. Implementation with Basic Architecture

A possible model of the role *Robot*, which is based on architecture described in Figure 6, is shown in Figure 8. The role checks actual distance of robot to the obstacle each 10 time units and offers information about robot's position by means of predicates *isClearRoad* and *isCloseToObstacle*. To get information about the distance, the role asks its subject by sending a message *getDistance*.

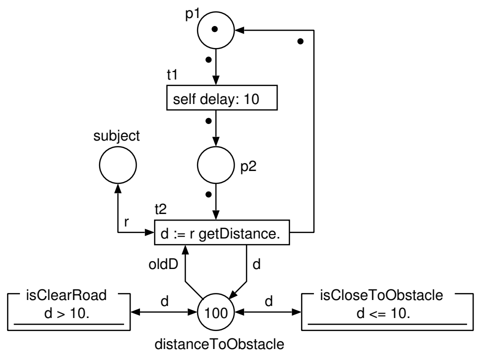


Figure 8. The role *Robot* – implementation for basic architecture.

We can exchange the simulated subject by real interface to the controlled robot. It is very simple—we only create instances of appropriate classes and do not care about used formalism. Figure 9 of Smalltalk code shows creating a subject as an instance of a Smalltalk class. This subject cooperates with a role and an activity modeled by OOPN. The object *Repos* represents the storage of all classes and simulations using OOPN or DEVS formalisms.

```
cAct  := Repos componentNamed: 'Scenario'.
cRole := Repos componentNamed: 'Robot'.
subj := RobotDevice new.
role := cRole forSubject: subjR.
actS := cAct forRole: roleR.
```

Figure 9. Accessing OOPN objects from Smalltalk.

Now, we demonstrate an accessing OOPN objects from product environment of Smalltalk. We send a command to start walking by means of a message `go`—the message passing is provided in the standard form. To test an object state, the predicates should be used. Since they are not ordinary methods, we have to access them in a special way. First, we obtain a special meta-protocol by sending a message `asPredicate`. Second, we can call synchronous port or negative predicate in the standard form of message passing. Third, the result represents a state of a called port/predicate, which has been tested. In our example, we test the predicate `isCloseToObstacle` and if the result is true, then we stop robot's walking by sending a message `stop`. The example is shown in Figure 10.

```
role go.
r := role asPredicate isCloseToObstacle.
r ifTrue: [ role stop ].
```

Figure 10. Message passing and predicate testing.

Of course, proposed solution is not sufficient for our case, because we need to test this condition until it becomes true. Therefore we can use one of following ways—to use waiting for specified condition or to define *a listener*. The first way is shown in Figure 11. We simply use a message `waitFor:` from the meta-protocol, which blocks until the specified condition becomes true, i.e., the port `isCloseToObstacle` becomes fireable.

```
role go.
role asPredicate waitFor: #isCloseToObstacle.
role stop.
```

Figure 11. Waiting for a condition.

Second way is shown in Figure 12. It uses a message `listener:for:` from meta-protocol to define a listener, which is activated if the condition becomes true, i.e., the port becomes fireable.

```
role go.
role asPredicate
      listener: self
      for: #isCloseToObstacle.
```

Figure 12. Setting a listener.

The activation of listener means that the special message `conditionSatisfied:` is sent to object, which is specified as a first argument. The example of its implementation is shown in Figure 13.

```
method conditionSatisfied: aCond
    (aCond == #isCloseToObstacle)
        ifTrue: [ role stop ].
```

Figure 13. Listener implementation.

B. Implementation with DEVS Architecture

Because the architecture changes, we have to modify classes describing system behavior. The component *Behavior* encapsulate OOPN model, which defines the class *Robot* as its initial class, so that ports are mapped to places of the *Robot* object net. This modified object net is shown in Figure 15. Place named *request*, resp. *answer*, corresponds to output port *request*, resp. input port *answer*.

The example of accessing DEVS components and their object interface is shown in Figure 14. First, we get a DEVS simulation named *R01*, which is based on architecture from Figure 7. Second, we obtain DEVS component *Behavior*, which is able to communicate through its ports. Nevertheless, this component is described by OOPN, so that it is possible to use object interface of its initial object (an instance of the class *Robot*) too. To get the object interface, we send a special message `objectInterface` from the component meta-object protocol.

```
s1 := Repos componentNamed: 'R01'.
cB := c1 componentNamed: 'Behavior'.
iB := cB objectInterface.
```

Figure 14. Obtaining object interface to the inital object.

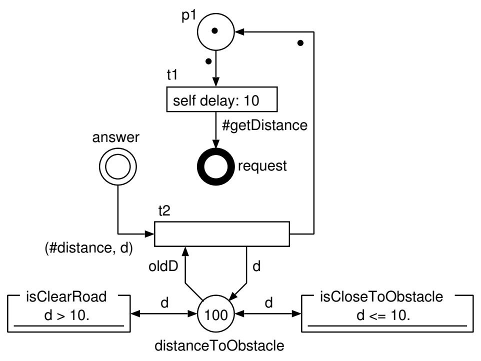


Figure 15. The role *Robot* – implementation for DEVS architecture.

VII. Conclusion and Future Work

The paper dealt with a possibility to deploy formal models to target application using specific application framework. It allows to use formal models as a basic design, analysis and programming means combining simulated and real components. The main advantage of that approach is no need for code generation and further investigation of deployed systems using the same formal models.

The proposed approach has one main disadvantage—usage of application framework, which interprets formal models directly demands of increased requirements on memory size and system performance. The future research will aim at efficient representation of choosed formal models and interoperability with another product environment. The application framework will be adapted to new conditions having lesser requirement for resources.

Acknowledgment

This work has been supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), by BUT FIT grant FIT-S-11-1, and by the Ministry of Education, Youth and Sports under the contract MSM 0021630528.

References

[1] S. Beydeda, M. Book, and V. Gruhn, *Model-Driven Software Development*. Springer-Verlag, 2005.

[2] M. Broy, J. Gruenbauer, D. Harel, and T. Hoare, Eds., *Engineering Theories of Software Intensive Systems: Proceedings of the NATO Advanced Study Institute*. Kluwer Academic Publishers, 2005.

[3] C. Raistrick, P. Francis, J. Wright, C. Carter, and I. Wilkie, *Model Driven Architecture with Executable UML*. Cambridge University Press, 2004.

[4] D. S. Frankel, *Model Driven Architecture: Applying MDA to Enterprise Computing*, ser. 17 (MS-17). John Wiley & Sons, 2003.

[5] R. Kočí and V. Janoušek, "System Design with Object Oriented Petri Nets Formalism," in *The Third International Conference on Software Engineering Advances Proceedings ICSEA 2008*. IEEE Computer Society, 2008, pp. 421–426.

[6] R. Kočí and V. Janoušek, "OOPN and DEVS Formalisms for System Specification and Analysis," in *The Fifth International Conference on Software Engineering Advances*. IEEE Computer Society, 2010, pp. 305–310.

[7] M. Češka, V. Janoušek, and T. Vojnar, *PNtalk — a Computerized Tool for Object Oriented Petri Nets Modelling*, ser. Lecture Notes in Computer Science. Springer Verlag, 1997, vol. 1333, pp. 591–610.

[8] R. Kočí and V. Janoušek, *Simulation Based Design of Control Systems Using DEVS and Petri Nets*, ser. Lecture Notes in Computer Science. Springer Verlag, 2009, vol. 5717, pp. 849–856.

[9] J. L. Fernandez, R. Sanz, E. Paz, and C. Alonso, "Using hierarchical binary Petri nets to build robust mobile robot applications: RoboGraph," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 1372–1377.

[10] C. Rust, F. Stappert, and R. Kunnemeyer, "From Timed Petri Nets to Interrupt-Driven Embedded Control Software," in *International Conference on Computer, Communication and Control Technologies (CCCT 2003)*, 2003.

[11] O. Bayo-Puxan, J. Rafecas-Sabate, O. Gomis-Bellmunt, and J. Bergas-Jane, "A GRAFCET-compiler methodology for C-programmed microcontrollers, In Assembly Automation," *Assembly Automation*, vol. 28, no. 1, pp. 55–60, 2008.

[12] R. Valk, "Petri Nets as Token Objects: An Introduction to Elementary Object Nets." in *Jorg Desel, Manuel Silva (eds.): Application and Theory of Petri Nets; Lecture Notes in Computer Science*, vol. 120. Springer-Verlag, 1998.

[13] D. Moldt, "OOA and Petri Nets for System Specification," in *Object-Oriented Programming and Models of Concurrency*. Italy, 1995.

[14] L. Cabac, M. Duvigneau, D. Moldt, and H. Rölke, "Modeling dynamic architectures using nets-within-nets," in *Applications and Theory of Petri Nets 2005. 26th International Conference, ICATPN 2005, Miami, USA*, 2005, pp. 148–167.

[15] V. Janoušek and R. Kočí, "PNtalk: Concurrent Language with MOP," in *Proceedings of the CS&P'2003 Workshop*. Warsaw University, Warsawa, PL, 2003.

[16] B. Zeigler, T. Kim, and H. Praehofer, *Theory of Modeling and Simulation*. Academic Press, Inc., London, 2000.

[17] A. GoldBerk and D. Robson, *Smalltalk 80: The Language*. Addison-Wesley, 1989.

[18] R. Kočí and V. Janoušek, "Modeling and Simulation-Based Design Using Object-Oriented Petri Nets: A Case Study," in *Proceeding of the International Workshop on Petri Nets and Software Engineering 2012*, vol. 851. CEUR, 2012, pp. 253–266.

[19] R. Kočí and V. Janoušek, "Specification of UML Classes by Object Oriented Petri Nets," in *ICSEA 2012, The Seventh International Conference on Software Engineering Advances*. Xpert Publishing Services, 2012, pp. 361–366.

A Measurement-based Approach to Software Development Process Tailoring in R&D Organization

Apinporn Methawachananont and Pawarat Nontasil
National Electronics and Computer Technology Center (NECTEC)
National Science and Technology Development Agency (NSTDA)
Pathumthani, Thailand
apinporn.methawachananont@nectec.or.th, pawarat.nontasil@nectec.or.th

Abstract— **In case of R&D (Research and Development) organization, the problems of SPI (Software Process Improvement) are focused on how to tailor the process properly because researchers always ask to take the least time and the most benefits for implementing established processes. Process tailoring strategy is a key to attract the researchers for applying the processes. It is a challenge for EPG (Engineering Process Group) to find out the best solution for the organization. EPG has to prove if the software development processes are suitable for the research. Measurable CSF (Critical Success Factors) and how to tailor appropriately influences the quality of the process.**

Keywords-Process Tailoring Strategy; Software Process Improvement; Engineering Process Group; Critical Success Factor

I. INTRODUCTION

Referring to Process Maturity Profile 2012 by SEI [1], many organizations have been struck at CMMI (Capability Maturity Model Integration) Maturity Level 3 because of missing quantitative project data; this is valid especially for government organizations, which tend to apply international standards for AEC (ASEAN Economic Community) opportunity. MA (Measurement and Analysis) is an important process area from all the 22 process areas which CMMI has specified and it affects to upgrade SPI in the organization. The problem is that there is no experience in this process. NECTEC tries to do research about it and expects to make the SMEs to understand better in MA.

Each software development project can have different SDLCs (Software Development Lifecycles) depending on its constraints that can be size, cost, effort, time, customer requirement, business/project goal, capability, culture, etc. There are various SDLCs including Waterfall Model, V-shaped SDLC, Structured Evolutionary Prototyping Model, RAD (Rapid Application Model), Incremental SDLC, Spiral SDLC, Agile SDLC, etc. Each SDLC has strengths and weaknesses which collect from past implementation [4]. But, each organization can adapt them to align with its optimizing processes like NECTEC where tailors Agile SDLC to be own SDLC called "Adaptive SDLC". Currently, agile methodology [18] is capturing more, especially the extreme method and a survey indicates percentage of companies which get better responses in main aspects such as 93% productivity, 88% quality, 49% cost and 83% business satisfaction [2]. Positive and Negative features from implementing agile methodology are identified in Figure 1.

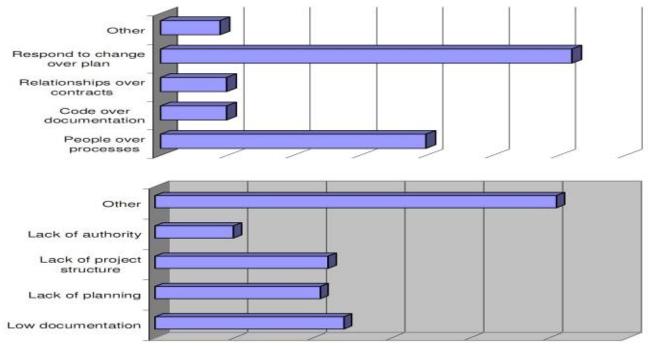
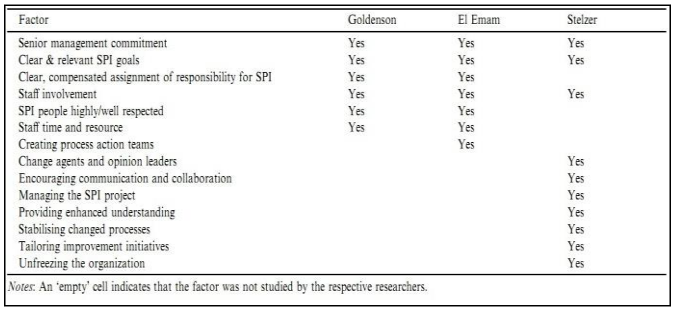


Figure 1. Positive and Negative features from implementing Agile [2]

Some perspectives for the organizational requirement of a metrics program have been classified [10]. Three main factors, which affect the SPI program, include senior management commitment, clear and relevant SPI goals, and staff involvement, as shown Figure 2.



Factor	Goldenson	El Emam	Stelzer
Senior management commitment	Yes	Yes	Yes
Clear & relevant SPI goals	Yes	Yes	Yes
Clear, compensated assignment of responsibility for SPI	Yes	Yes	
Staff involvement	Yes	Yes	Yes
SPI people highly/well respected	Yes	Yes	
Staff time and resource	Yes	Yes	
Creating process action teams		Yes	
Change agents and opinion leaders			Yes
Encouraging communication and collaboration			Yes
Managing the SPI project			Yes
Providing enhanced understanding			Yes
Stabilising changed processes			Yes
Tailoring improvement initiatives			Yes
Unfreezing the organization			Yes

Notes: An 'empty' cell indicates that the factor was not studied by the respective researchers.

Figure 2. Factor affecting to SPI program [11]

Seven advantages of Measurement are identified in Rational Edge article. They include 1) Improve visibility, 2) Communicate effectively, 3) Identify and correct problems early, 4) Make key trade-off, 5) Track specific project objectives, 6) Manage risks, and 7) Defend and justify decisions and plan future projects. However, it is hard to establish measures because of no having certain set for all organizations. It depends on their strategy, technology, and the route of competition.

First an overview of measurement-based methodology is provided. Then the paper presents a result of implementing measurement in R&D organization and how to work with MA process. The CSF for the MA implementation is identified. Finally, an effort to find out the better measure for R&D work is proposed.

II. METHODOLOGY

A. *Measurement and Analysis Process based on CMMI*

Measurement and Analysis (MA) process area is grouped in support category. Its objective is to develop and maintain measurement capability for supporting management information needs. There are 2 specific goals; each goal consists of 4 practices to fulfill the goal. The goals are to align measurement and analysis activities and to provide measurement results. CMMI just guide what to do so each organization has to find out how to do the best. Each organization can have different MA process depending on its goal. The process can be changed periodically because the organization can change its goal. Figure 3 shows the relationship between MA process and other processes.

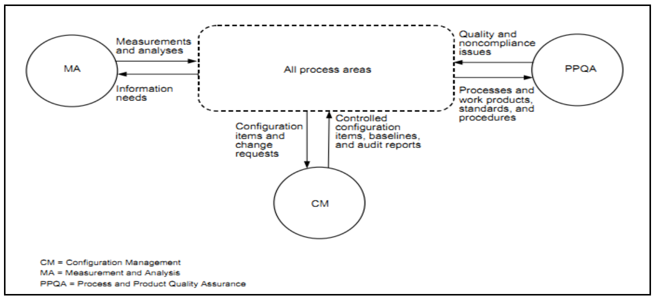

MA
Measurements and analyses
Information needs
All process areas
Quality and noncompliance issues
Processes and work products, standards, and procedures
PPQA
Configuration items and change requests
Controlled configuration items, baselines, and audit reports
CM
CM = Configuration Management
MA = Measurement and Analysis
PPQA = Process and Product Quality Assurance

Figure 3. The relationship between MA process and others [14]

B. *MA Process Evaluation Approach*

The measurement management in organization has several methodologies. The Software Engineering Institute (SEI) published an interesting method called "Measurement and Analysis Infrastructure Diagnostic Method (MAID)". MAID guides the organizations to evaluate key characteristics of their measurement programs [15]. This method is based on criterion. A set of criteria for evaluating each MA process has been introduced in [15]. The MAID method has four phases comprising (1) Measurement Planning, (2) Data Collection and Storage, (3) Data Analysis, and (4) Measurement Reporting. The criteria are implemented by evaluation team in the 2nd and 3rd phase. We tried to apply MAID method to appraise the CMMI-based MA process. Some activities have been selected to be implemented, such as Review MA documents, Conduct interviews and examinations, etc. However, the criteria cover various issues including data analysis, reporting, process documentation, etc. Another interesting approach is called "Standard CMMI Appraisal Method for Process Improvement (SCAMPI)", which supports evaluation of CMMI-based process in term of opportunity for improvement (OFI). There are A, B and C types; SCAMPI A is the official appraisal and others will reduce strictness, respectively. Figure 4 presents an appraisal direction [16].

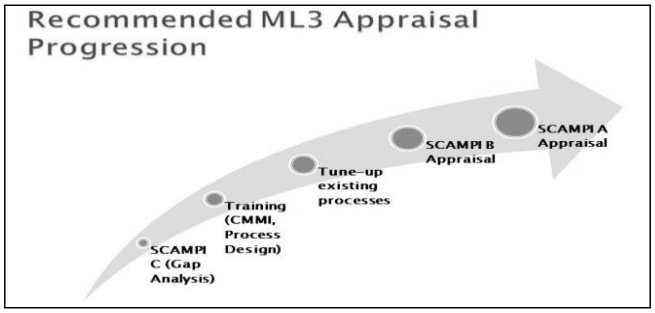

Recommended ML3 Appraisal Progression
SCAMPI C (Gap Analysis)
Training (CMMI, Process Design)
Tune-up existing processes
SCAMPI B Appraisal
SCAMPI A Appraisal

Figure. 4. An example of appraisal method [17]

C. *Goal-Question-Metric Paradigm (GQM)*

Goal-Question-Metric Paradigm is invented Basili [19] from the University of Maryland College Park and Software Engineering Laboratory at the NASA Goddard Space Flight Center. This approach is based on the idea of goal-oriented measurement. In Figure 8, we apply GQM approach to analyze the measures. We started with analyzing the organizational goals, which came from the executive policy and found out related measures leading to achieve those goals via a set of questions. GQM approach can divided into three levels, as shown in the Figure 5:

- Conceptual Level (Goal): We set up Business Goals that is the goal in the measurement goals.
- Operational Level (Question): We define a set of questions to achieve the goal.
- Quantitative Level (Metric): We formulate the measure to answer the question in Operational Level.

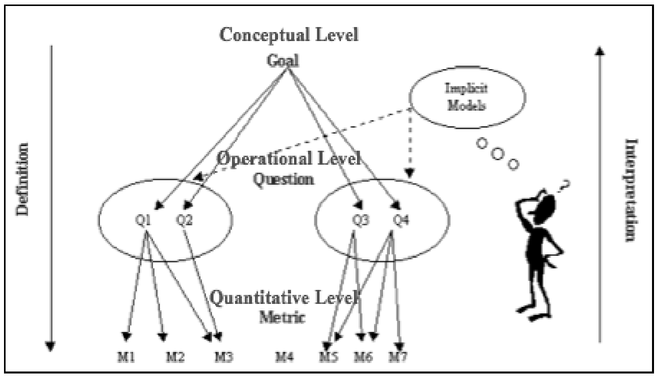


Figure 5. GQM Levels [13]

III. MEASUREMENT IMPLEMENTATION

A. *A Set of Measures for R&D Organization*

Primary quality attributes, which impact achievement of the SPI program, are summarized in 5 categories involving performance, stability, compliance, capability, and improvement [7]. Organizational Metrics are aligned with these categories to specify their values. For Project Level, there are different quality attributes categories and supporting metrics. An example of metrics in each category is shown in Table I.

TABLE I. AN EXAMPLE OF METRICS IN QUALITY CATEGORIES [7]

Level	Quality Category	Example of Supporting Metrics
Organization	Performance	Completeness of requirement, Resource utilization versus the plan
	Stability	Effectiveness of Scope, schedule, and cost-tracking processes
	Compliance	Product conformance with requirement, # workarounds required
	Capability	Use of knowledge, skills, and competency profiles
	Improvement	Involvement of individual team members initiatives, Effect of technology in terms of performance improvement
Project	Resource	Cost/budget, Resource Utilization
	Progress	Development progress, Incremental capabilities performance
	Technical	Requirement stability, Design stability, Error margins
	Quality	Defects, Rework, Defect removal rate
	Productivity	Cost performance index, Trends in cost, schedule, efficiency
	Completion Activity	Quality gate task status, Quality gate passed
	Change	Percent change to product baseline per period
	Staff	Percent voluntary staff turnover, Percent overtime
	Risk	Risk impact and reduction, Risk Liability, Anonymous warning

The appropriate measures depend on the organization's strategy, technology, and economic situation [7]. From a survey, top 10 project measures consist of ROI (Return on Investment), Productivity, Cost of Quality, Cost of Performance, Schedule Performance, Customer Satisfaction, Cycle Time, Requirements Performance, Employee Satisfaction and Alignment to Strategic Business Goals [7]. Figure 6 and Table II present some characteristics of R&D works leading to different measures.

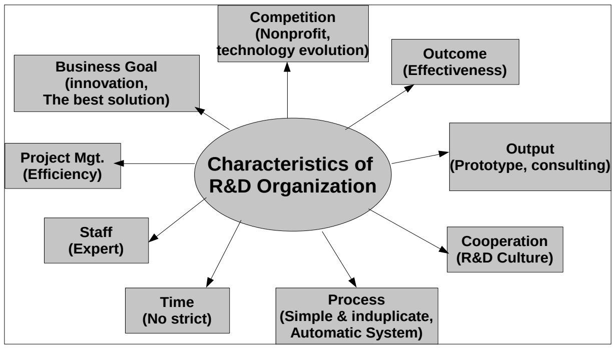

Competition (Nonprofit, technology evolution)
Business Goal (innovation, The best solution)
Outcome (Effectiveness)
Project Mgt. (Efficiency)
Characteristics of R&D Organization
Output (Prototype, consulting)
Staff (Expert)
Cooperation (R&D Culture)
Time (No strict)
Process (Simple & induplicate, Automatic System)

Figure 6.Characteristics of R&D Organization

TABLE II. MEASURES/METRICS SUPPORTING R&D WORKS [7]

Category	Characteristic	Measures/Metrics
Business Goal	Research an innovation	- Improvement Trends/ Pattern - Operational Trends/ Patterns - Alignment to Strategic Business Goals
	find out the best solution	Customer Satisfaction, # problems reduced
Competition	Nonprofit	% research linked to business unit or corporate strategic planning, R&D as a % sales
	Compete with technology evolution	#ideas, #inventions submitted, #patents challenged
Process	Simple and induplicate	Customer Satisfaction
	Supporting Automatic System	% process operated automatically
Staff	expert	% R&D staff with related experience
Time	No strict in time	R&D time variance vs. budget
Outcome	effectiveness	Return On Investment, Work satisfaction, etc.
Output	Lab prototype	Productivity
	Consulting SMEs to upgrade product	Customer Satisfaction, market share
Cooperation	R&D Culture depends on behavior of researcher	Employee Satisfaction
Project Management	Efficiency, Different between planned and actual values	Cost Performance, Schedule Performance

It is impossible to record all data to respond the related measurement. Thus, the organization should consider the measures from the needs of the executive. How to get data supporting all measures for R&D works has many channels such as GQM, MAID, CMMI, Lesson learned, etc.

Figure 7 presents three types of indicators including success indicators, progress indicators, and analysis indicators [11]. EPG can apply this idea to find out the measures.

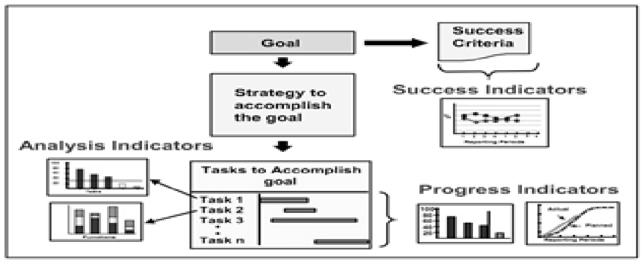
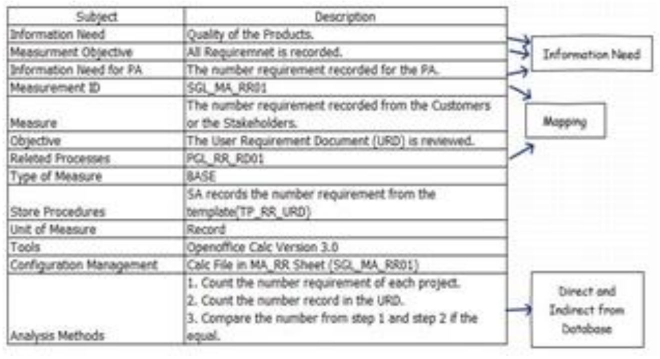


Figure 7. Types of indicators [11]

B. *Implementation in R&D Unit*

Generally, many organizations including NECTEC start to follow Specific Goals and Practices of Measurement Process Area. NECTEC's EPG established a lot of data for achieving measurement goals but finally users could not record all established inputs because they needed a lot of effort (to understand, to record, to attend, etc.). Moreover, the recorded data was not correct because they usually recorded after related activities had occurred although there were templates to support them completely. Finally, the process improvement program could not achieve measurement activities.

Figure 8 presents mapping the organizational needs to related MA processes and established analysis methods. Table III shows the lesson learned from NECTEC's CMMI implementation (2010-2011) including its strengths and weaknesses. Each role in a project has to record data for supporting the measurement process. There are 10 different templates for project manager to input the data which depends on applied processes. Figure 9 shows an example of MA templates and Table IV proposes the new information needs and how to obtain the best measures for NECTEC or R&D organization comparing strategy from CMMI and GQM.



Subject	Description
Information Need	Quality of the Products.
Measurment Objective	All Requiremnet is recorded.
Information Need for PA	The number requirement recorded for the PA.
Measurement ID	SGL_MA_RR01
Measure	The number requirement recorded from the Customers or the Stakeholders.
Objective	The User Requirement Document (URD) is reviewed.
Releted Processes	PGL_RR_RD01
Type of Measure	BASE
Store Procedures	SA records the number requirement from the template(TP_RR_URD)
Unit of Measure	Record
Tools	Openoffice Calc Version 3.0
Configuration Management	Calc File in MA_RR Sheet (SGL_MA_RR01)
Analysis Methods	1. Count the number requirement of each project. 2. Count the number record in the URD. 3. Compare the number from step 1 and step 2 if the equal.

Figure 8. An example of MA Analysis

TABLE III. LESSON LEARNED FROM IMPLEMENTATING MEASUREMENT PROCESS OF NECTEC (2010-2011)

Needs	#Way to record the Measure by each role	Lesson learned		Suggestion
		Strengths	Weaknesses	
Progress of the project	PM: 10 CM: 1 SA: 4 Dev: 3 Rev: 3 Tester: 1	- all PAs covering measures - Having data to respond all related measures - Recorded by related roles	-no automatic record -spend time to record -often forget to record -a lot of data to record -no use all data -no understand clearly	1. no MA experience 2. no need to record all data initially 3. too difficult to record 4. duplicate record 5. no align with real activities 6. no need to record some measures (get ROI from responsible unit) 7. join with QA or PM to support MA records
Quality of the project	QA: 1 Cus: 1			
ROI of the project	PM: 1			
Others	EPG: 3			

Remarks:
-1st deployment
>>fail (no complete data, no right data, no record immediately, etc.)
-2nd deployment
>>almost fail (some measures are selected to respond some needs but not be satisfied by the stakeholder)

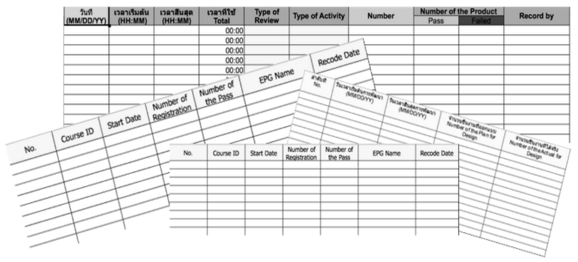


Figure 9. An example of MA templates

C. *Lesson Learned from MA Implementation in R&D Unit*

The lesson learned from the past implementations makes us understand more about the importance of MA process. Many problems occurred in the MA implementation period as shown in Figure 10. The problems and their solutions are summarized for the next implementation in Table IV.

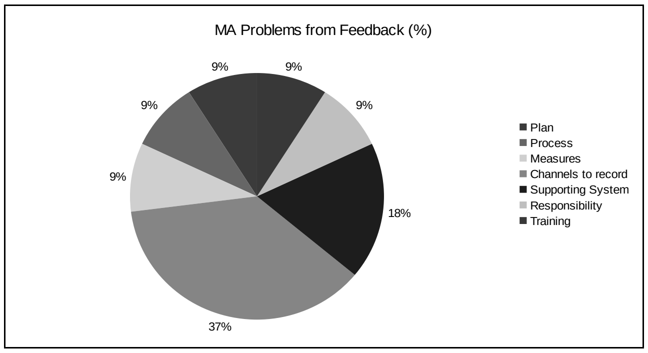


Figure10. MA Problems in SPI Implementation Period (%) [5]

D. Tailoring the Process

There are two levels for tailoring software development process. Firstly, it focuses on organizational process which NECTEC's EPG tries to optimize Software Development Lifecycle (SDLC) models from brainstorming of stakeholders. There are other related processes, such as training process, improvement process, etc. They have to be consistent with organizational policies and goals. Another level is to tailor processes and products in each project. How can one know that each tailoring can respond the information needs completely? A product tailoring template should be established for stakeholder who requests to do other products instead. What is the best criterion for tailoring the process and the product responding to organizational goals? Currently, EPG has to examine each tailoring in each project. If you want to focus on quality of process improvement works, you also have to realize appropriate conditions for tailoring the product. However, alternatives should be considered.

IV. OPTIMIZING MEASURES FOR R&D ORGANIZATION

A. Measures in R&D Organization

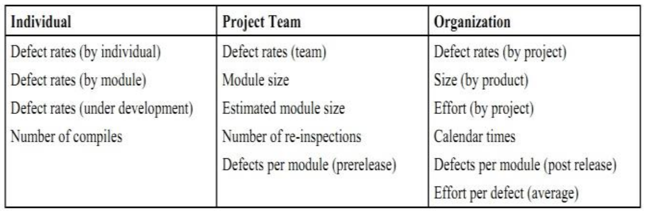
This paper presents NECTEC to be representative of R&D organization. We start from current business goals as follows. The measures which come from GQM and survey result are identified as follows:

- Tracking of the project: Milestone completion, Resource utilization, Risk impact and reduction, Project Completions per year, Number of active/on –hold/closed projects, periodically.
- Quality of the project: Product defects, Defects by activity, Deviation from standard.
- Error/fault/failure rates, Product failures, Customer complaints.
- Return on Investment (ROI): Investment in R&D/Project Cost, Evaluated benefits from applying related products periodically, Comparison between cost and evaluated benefits, Customer satisfaction/ dissatisfaction, Customer Retention.
- Engineering Excellence: Depth width and knowledge, Skills and productivity, Building character to perform within moral and ethical framework.

TABLE IV. MA PROBLEMS AND SOLUTIONS

#	MA Aspect	MA Problem	% Feedback (project and appraisal team)	Proposed MA Solution
1	Plan	Don't know why to do MA process	9%	-Clear MA plan and inform to stakeholder
2	Process	Incorrect Steps to record MA leading wrong data (some records)	9%	-clear understanding of the advantage from MA data. -executive supporting policy
3	Measure	Too much for responding the organization 's needs	9%	-Apply GQM methodology to identify the measures (traceability) -Start small and showcase a success
4	Channels to record	Too hard to record	37%	-Access rapidly and easily -Simple Templates and not many templates.
5	Supporting System	No application to support MA process	18%	-Retrieve data from operation automatically
6	Responsibility	No assign the person to track,collect, analyze, summarize, and report all MA records	9%	-Assign a person to track and collect all MA records periodically
7	Training	Forget step to record MA.	9%	-clear understanding of the type and purpose of each indicator -simple guideline to remind MA process/step

NECTEC is implementing these measures for organizational level. The MA result has to respond the executive's information needs or policies. However, measures in the project level can use some measures from the organizational level and add some measures which impacts to achieve the project goals such as measures proposed to the 1st-2nd business goals. Besides the tailoring the process also has to support the established measures especially in project aspect. Figure 11 shows an example of duplicated measures in different aspects [6].



Individual	Project Team	Organization
Defect rates (by individual)	Defect rates (team)	Defect rates (by project)
Defect rates (by module)	Module size	Size (by product)
Defect rates (under development)	Estimated module size	Effort (by project)
Number of compiles	Number of re-inspections	Calendar times
	Defects per module (prerelease)	Defects per module (post release)
		Effort per defect (average)

Figure 11. An example of duplicated measures in different aspects

B. Tailoring Criteria for R&D Organization

Another important activity which needs measures properly for implementing the project is SW development Process Tailoring. Five main causes enforce EPG to tailor the process including resource, communication, requirement management, political and technical [8]. The process includes all related documentation such as SDLC, template, guideline, etc. Concerning the lesson learned, EPG should tailor the SDLC covering all types of the R&D projects. Currently, NECTEC has tailored the SDLC into 3 types involving waterfall, rapid prototyping, and adaptive models. Each model has different documents that authorized person can request to tailor the documents with his/her reasons. EPG will consider the requests in 2 aspects, which cover related standard goals and established measures.

Another challenge issue needed is to find criteria for choosing the appropriate process (global process model and methods and tools supporting those models), evaluating its suitability and improving it continuously [9]. Referring to the characteristics of R&D works, the measures should be established to evaluate its consistency with the information needs. There are two tailoring level including organization and project levels. The tailoring approach is one of the improvement methodologies. Purpose of tailoring the process in a project is data collection to indicate all candidates of process and work product in R&D work. Error, fault and failure analysis are selected to respond the tailoring purpose. Furthermore, measures which should be also realized for tailoring the process effectively include coverage attribute following the standard process and established measures. EPG has to consider quality in coverage for tailoring both process and work products. The criteria supporting EPG to validate the tailored process is proposed as follows:

- Measures, which respond the organization/project goals from tailoring processes, are still recorded.
- Measures, which respond the organization/project goals from tailoring products are still recorded.
- Tailoring Processes still respond to organization/project goals comparing with default processes.
- Tailoring Products still respond to organization/project goals comparing with default products.
- The process (including related products) still responds to established requirements.

V. CONCLUSION AND FUTURE WORK

This paper presented how to implement MA process in R&D organization and proposes an idea to improve it including measure analysis and tailoring conditions. To apply international frameworks can make officers work professionally. The R&D organization has specific business goal which impacts to establish the measures for indicating quantified improvement level. Tailoring the process is a measure which supports flexible process. How to tailor the process effectively needs to be analyzed systematically.

A set of measures has to adjust in parallel with changed business goals. Moreover, supporting data should be recorded automatically and should not be operated repeatedly by project team. It is a challenge for the next research to refine better processes and measures by analyzing actual result continuously and make them more generic and systematic for distributing their advantages to others.

REFERENCES

[1] Software Engineering Institute, "CMMI for SCAMPI Class A Appraisal results 2012 Mid-Year Update," Carnegie Melon, Pittsburgh, September 2012.

[2] M. Johnson, "Agile Methodologies Survey Results," A Passion For Excellence, Shine Technologies Pty. Ltd, 2003.

[3] C. Ebert, R. Dumke, M. Bundschuh, and A. Schmietendorf, "Best Practices in Software Measurement," Springer-Verlag berlin Heidelberg, 2005.

[4] Y. Berra, "Software Development Life Cycle (SDLC)," presentation.

[5] Software Engineering Labolatory, "Research and Development Report for NECTEC-CMMI phase 1," NECTEC/NSTDA, 2009.

[6] W. Goethert and W. Hayes, "Experiences in Implementing Measurement Programs," Copyright by Carnegie Mellon University, November 2001.

[7] Center for Business Practices, "Measures of Project Management Performance and Value". PA 19083 USA.

[8] P. Xu and R. Balasubramaniam, "Using Process Tailoring to Manage Software Development Challenges," IEEE Computer Society, 2008.

[9] V. R. Basili and H. D.Rombach, "Tailoring the Software Process to Project Goals and Environment," ACM, 1987.

[10] D. N. David, T. Hall, and N. Baddoo, "A framework for evaluation and prediction of software process improvement success," University of Technology in Australia and University of Hertfordshire in UK, 2001.

[11] A. Rainer and T. Hall, "Key success factors for implementing software process improvement: a maturity-based analysis," The Journal of Systems and Software 62, 2002, pp. 71-84.

[12] Royal Academy of Engineering, “Measuring Excellence in Engineering Research,” London, January 2000.

[13] G. Xiaodong and M. Li, “Organization Application Oriented Software Process Measurement Model,” ISCSCT -2008-80,2008.

[14] Software Engineering Institute, “CMMI for Development, version 1.3,” Technical Report, November 2010.

[15] Software Engineering Institute, “Measurement and Analysis Infrastructure Diagnostic (MAID) Evaluation Criteria, version 1.0,” Technical Report, December 2009.

[16] Software Engineering Institute, “Standard CMMI Appraisal Method for Process Improvement (SCAMPI^{SM}) A, Version 1.3: Method Definition Document,” Handbook, March 2011.

[17] J. Dalton, “What are the steps to achieving a Maturity level of CMMI? ,” http://askthecmmiappraiser.blogspot.com, February 2012. [retrieved: May, 2013]

[18] J. A. H. Alegria and M. C. Bastarrica, “Implementing CMMI using a Combination of Agile Methods,” Clei Electronic Journal, Volume 9, Number 1, Paper 7, June 2006.

[19] V. Basili, “Software Modeling and Measurement: The Goal/Question/Metric Paradigm,” College Park, MD, USA: University of Maryland, Technical Report CS-TR-2956, 1992.

Towards Probabilistic Models to Predict Availability, Accessibility and Successability of Web Services

Abbas Tahir, Sandro Morasca, Davide Tosi
Department of Theoretical and Applied Sciences
Università degli Studi dell'Insubria
Varese, Italy
tahir_a@acm.org, {sandro.morasca, davide.tosi}@uninsubria.it

***Abstract*— Web Services are gaining increasing attention as programming components and so is their quality. The external qualities of Web Services (i.e., qualities that are perceived by their users) such as the OASIS sub-quality factors *Availability*, *Accessibility*, and *Successability* can only be measured at late stages after the deployment and the provisioning of the Web Service. This may necessitate expensive rework if the targeted levels of qualities are not satisfactorily met. A reliable prediction of the values of the external qualities at early phases during development may totally remove the need for reworking and hence save valuable resources. In this paper, we describe an approach for building and empirically evaluating probabilistic prediction models for the Web Services external sub-quality factors *Availability*, *Accessibility*, and *Successability* based on internal static and dynamic quality measures (e.g., Cyclomatic Complexity and Distinct Method Invocations). A methodology was established that involves the collection of a set of predefined quality measures and then performing regression analysis to identify any correlation between them and the above mentioned external qualities. For this purpose, a framework for data collection and evaluation was designed, implemented and tested. The results of the preliminary evaluation of the framework showed that it is feasible to collect all the data points necessary for the regression analysis and model building activities. We are currently working towards adding about 18 more Web Services to our testbed in order to carry out a wider controlled experiment and then to build possibly accurate probabilistic prediction models for *Availability*, *Accessibility*, and *Successability*.**

Keywords-quality models; web services; measurement; metrics; probabilistic models; quality prediction

I. INTRODUCTION

Web Services (WSs) are gaining more attention as programming components for different software applications. They play an important role in service-oriented architectures where loosely coupled programming components or services deliver their functionality over a network – often over the Internet. The quality of such architectures depends heavily on the quality of its individual service components, which are usually WSs. Therefore, the quality of WSs is becoming a major concern. Users of WSs are usually careful (among others) about the *availability* of WSs they are relying on. They also need to know whether the WSs are *accessible* (i.e., they actually accept requests) while available and whether they *successfully* deliver responses for the incoming requests. These concerns are referred to as the *Availability*, *Accessibility*, and *Successability* of WSs.

The OASIS Web Services Quality Model (WSQM) Technical Committee [1] is currently working towards a quality model for WSs. The committee developed specifications for WSs quality factors (WSQF) [3] that cover the development, usage and management of WSs. One of the quality factors described in the specification is the Service Level Measurement Quality that consists of sub-quality factors including *Availability*, *Accessibility*, and *Successability*.

All of the above mentioned quality factors are considered external software quality measures according to the definition provided in the ISO/IEC standard 25000 [4]. On the other hand, internal software quality measures [4] are those measures concerned with the static attributes of software products (e.g., number of lines of code). Such measures are usually related to the software architecture and design and do not require the execution of the targeted software. Measures that can only be collected by executing the software are called dynamic measures. For example coupling between class objects CBO is a well-known static quality measure. If it is measured in runtime, it is called dynamic coupling between objects DCBO and considered as a dynamic software quality measure.

The external quality measures *Availability*, *Accessibility*, and *Successability* of a WS can be only measured when the WS is already developed, deployed and exposed to users. If these external quality measures can be predicted early during the development phase, they can provide valuable information that may positively influence the engineering of WSs with regards to the three sub-quality factors.

Other researchers worked towards predictive models for software quality. Dragan Ivanovic et al. [5] proposed a methodology for predicting Service Level Agreement (SLA) violation during service composition at run-time. They used the structure of the composition and properties of the component services to derive constraints to model SLA conformance and violations. These models are used for predicting satisfaction and violation of the constraints in a specific scenario. Xing et al. [6] proposed an approach to predict software quality by adopting support vector machine

(SVM) in the classification of software modules based on complexity metrics.

There are many factors that may influence the time-related behavior, and therefore some external qualities of the WS (e.g., network, hardware, application server, application software, etc.). In this research, we are focusing on the WS's application software since in a typical WSs development project, only the WS's logic is implemented and all the other elements are not developed but only used for deployment and hosting purposes. Factors other than the WS's application software are isolated by using similar configurations for all WSs under test. Our aim is to help predicting external qualities in early stages of WSs development projects based on static internal quality measures as well as the internal dynamic behavior of WS's application software measured through different dynamic measures.

In this paper, we present a framework for (1) collecting some static and dynamic quality measures from WSs, and (2) applying statistical approaches to identify any correlation between the static and dynamic measures collected and WSs *Availability*, *Accessibility*, and *Successability*, and (3) the development of probabilistic models for the prediction of WSs *Availability*, *Accessibility*, and *Successability* based on the theory provided in [7].

The rest of this paper is structured as follows. Section II provides the necessary background by introducing the basic concepts and the theoretical basis on which this work is based. In Section III, the aims and objectives are introduced and the two main research questions are clearly stated. Then, the methodology followed and the data required for experimentation and model building are introduced (Section IV). A detailed technical description of the framework used for collecting necessary data during experimentation is provide in Section V. The results of short tests performed to build confidence on the framework are listed in Section VI. Finally, in Section VII, conclusions are drawn and future work plans are introduced.

II. BACKGROUND

In this section, we introduce the WSs quality factors defined by OASIS with focus on *Availability*, *Accessibility* and *Successability*; we discuss the theoretical background that is at the basis of our framework to compute external quality factors; we show the logistic regression approach that helps us in predicting external quality factors starting from the observation of internal quality metrics.

A. OASIS Web Services Quality Factors

As a result of the increased acceptance and utilization of WSs as programming components, the OASIS [2] standardization body established a technical committee [1] to define a quality model for WSs (WSQM). The model is centered around the identified WSQFs [3]. The quality factors are based on the functional and non-functional properties of the WSs. They are classified into 6 categories: Business value quality, service level measurement quality, interoperability quality, business processing quality, manageability quality, and security quality. Each category contains different related sub-quality factors. Service level measurement quality is subdivided into five sub-quality factors including *Availability*, *Accessibility*, and *Successability*.

Availability is defined as "a measurement which represents the degree of which web services are available in operational status. This refers to a ratio of time in which the web services server is up and running. As the DownTime represents the time when a web services server is not available to use and UpTime represents the time when the server is available, Availability refers to ratio of UpTime to measured time "

$$Availability = 1 - \frac{Down\,Time}{Measured\ Time} \tag{1}$$

Accessibility "represents the probability of which web services platform is accessible while the system is available. This is a ratio of receiving Ack message from the platform when requesting services. That is, it is expressed as the ratio of the number of returned Ack message to the number of request messages in a given time."

$$Accessability = \frac{number\ of\ \ Acks\ rmessages}{number\ of\ \ request\ messages} \tag{2}$$

Successability "is a probability of returning responses after web services are successfully processed. In other words, it refers to a ratio of the number of response messages to the number of request messages after successfully processing services in a given time. 'Being successful' means the case that a response message defined in WSDL is returned. In this time, it is assumed that a request message is an error free message."

$$Successability = \frac{number\ of\ \ response messages}{number\ of\ \ request\ messages} \tag{3}$$

B. Theoretical background

Morasca [7] introduces a probability-based approach for measuring the external qualities of software. The main assumption is that external qualities can be quantified by means of probabilities. The author proposes that "external software attributes should not be quantified via measures, but via probabilistic estimation models." This implies that instead of measuring the external qualities after the deployment and the exposure of a WS, we can predict them using probabilistic models.

Additionally, the introduced probability-based approach is rooted in the "probability representations", which are part of the well-founded Measurement Theory. Probability representations "has not yet been used in Software Engineering Measurement" [7].

Based on this theory, probabilistic models for different software external qualities models can be built. However, the accuracy of the models need to be assessed by carrying out empirical studies.

C. Logistic regression

Logistic regression [8] is a statistical analysis approach for predicting the outcome of dependent variables based on one or more independent variables.

The logistic regression curve (Fig. 1) is the graphical representation of the logistic function that can be expressed as follows for one independent variable x:

$$P(x) = \exp(\beta_0+\beta_1 x)/(1 + \exp(\beta_0+\beta_1 x)) \quad (4)$$

$$logit(x) = \beta_0+\beta_1 x \quad (5)$$

$P(x)$ is the probability that the dependent variable equals 1 for the independent variable x. β_0 and β_1 are the regression coefficients.

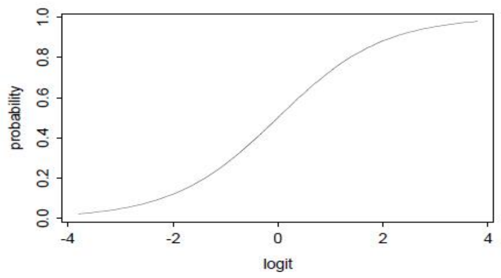


Figure 1. The logistic regression graph

As it is clear from Fig.1, the values of the dependent variable (probability) range from 0 to 1. Re-examining the formulas for calculating the external qualities *Availability*, *Accessibility*, and *Successability* presented in Section II, we can conclude that the resulting values of any of the three qualities range also from 0 and 1. Therefore, we intend to use the logistic regression in our analysis to identify any possible correlation between the internal and the external quality measures in order to facilitate the prediction of external quality factors starting from the static and dynamic measure of internal quality factors.

III. OBJECTIVES AND RESEARCH QUESTIONS

The main final aim of the work described in this paper is to develop probabilistic models for the quantification of the software sub-quality factors *Availability*, *Accessibility* and *Successability* identified in the OASIS WSQF [3] based on the theoretical basis provided in [7]. These models may predict the values of the above-mentioned factors in early phases (design-time and deployment-time), thus allowing for early adjustments during the development to satisfy any imposed requirements with regards to the three sub-quality factors. Additionally, knowing the need of adjustments in advance may also facilitate early evaluation of the impact (costs, human resources, etc.) for implementing the adjustments.

Our Objectives (O) can be summarized as follows:

- O1 - To build significant probabilistic models for the sub-quality factors *Availability*, *Accessibility* and *Successability*;
- O2 - To empirically evaluate the accuracy of the probabilistic models.

To achieve our objectives, we formulated the following research questions (RQ):

- RQ1 - Is it possible to build statistically significant probabilistic models for the WSs sub-quality factors *Availability*, *Accessibility* and *Successability*?
- RQ2 - How accurate are these models?

To build and empirically evaluate the probabilistic prediction models, we designed and implemented a framework able to support developers of WSs to collect and calculate metrics automatically and to measure external qualities.

IV. EXPERIMENTATION APPROACH

For model building and evaluation, we need to perform experimentation using a set of WSs. The approach we follow can be summarized as follows:

1. Selection of suitable WSs for experimentation;
2. Identification and selection of related software measures to be collected besides the external qualities *Availability*, *Accessibility*, and *Successability*;
3. Development of a framework for collecting the selected quality measures;
4. Data collection;
5. Analysis of the collected data and building probabilistic models for the external qualities *Availability*, *Accessibility*, and *Successability*.

The experimentation will be carried out as a controlled experiment, where (graduate) students will be used to interact with the WSs to collect quality measures.

A. *Web services selection*

The WSs needed for experimentation are selected based on the following criteria:

- Full access to the source code and the documentation of the WS to facilitate the evaluation of static and dynamic quality factors;
- The WSs are built using Java programming language, due to the fact that our framework is currently able to analyze Java components only;
- The WS provides the claimed functionality itself and it is not a "wrapper" for other services.

Since open source applications usually satisfy the above criteria, we focused on them.

Unfortunately, the process of identifying and selecting WSs satisfying all the aforementioned criteria ended with the availability of just one WS only. Specifically, we discovered and used as case study a WS released by Yesiltepe Softwareentwicklung [9], which satisfies all the above conditions. This WS provides a registry for artists. One issue with this WS is that the data of artists are stored on plane operating system files. This makes the application slow and not stable enough for concurrent accesses. Therefore, we modified the original WS to make use of an embedded database instead of plane files.

To overcome the limitation in the number of available Open-WSs on the net, we decided to manually convert free and open source Java applications into WSs (i.e., the functionalities provided by the Java applications are exposed on the Web). To perform this conversion, we used the Apache Axis2 framework [10]. For instance, we converted the application code2web [11], a utility application that converts Java source code into HTML, into a WS. For uniformity, we used the Axis2 framework to expose the functionalities of all the WSs selected for the experimentation.

To provide a statistically relevant set of WSs, we targeted at least 20 WSs for the complete experimentation of the approach. This process is an ongoing work so, in this paper, we focus on the experimental results of the two above-mentioned case studies.

B. Identification and selection of software measures to be collected

Building probabilistic models for the sub-quality factors *Availability*, *Accessibility*, and *Successability* involves the identification of the *dependent* variables and the (possibly) related *independent* variables. Since we aim to predict the sub-quality factors *Availability*, *Accessibility* and *Successability*, they are considered the dependent variables. The independent variables on which the prediction of the dependent variables depends are the software internal static and dynamic measures listed below. The static quality measures selected are well-known a widely accepted measures taken mainly from [12]. We also considered the dynamic behavior of the Web Services by including four dynamic metrics.

- **Static software measures:**
  - Lines of Code (LOC) is the number of lines of code in the WS's source code. It is a size measure that is usually used to assess the complexity of the software.
  - McCabe Cyclomatic Complexity (CC) counts the number of linearly independent paths in the WS's source code.
  - Weighted Methods per Class (WMC) is the sum of the McCabe Cyclomatic Complexity of all class methods.
  - Lack of Cohesion of Methods (LCOM) “is the number of pairs of methods in a class that don't have at least one field in common minus the number of pairs of methods in the class that do share at least one field. When this value is negative, the metric value is set to 0.” [13]
  - Afferent Couplings (Ca) is the number of other packages that depend upon classes in a specific package.
  - Efferent Couplings (Ce) is the number of other packages that the classes in the package depend upon.
  - Instability (I): The ratio of efferent coupling (Ce) to total coupling (Ce + Ca)
- **Dynamic software measures:**
  - Distinct Classes (DC) is “the count of the distinct number of classes that a method uses within a runtime session.” [14]
  - Dynamic Coupling Between Objects (DCBO) is the number of distinct classes a specific class is coupled to at runtime.
  - Object Method Invocations (OMI) is the total number of distinct methods invoked by each method in each object within a runtime session
  - Distinct Method Invocations (DMI) is “the count, within a runtime session, of the total number of distinct methods invoked by each method in each object.” [14]

Each data point for a specific WS in the regression graph is composed of two elements, the dependent variable (Y-Axis) and the independent variable (X-Axis). For example, suppose that the measured A*vailability* value is 0.922 and WMC value is 7.60, then (7.60, 0.922) is a data point on the regression graph.

C. Data Collection.

The static software measures (e.g., LOC and WMC, etc.) are first calculated for all WSs using two different tools, namely, CodePro AnalytiX [15] and the Eclipse Metrics plugin [16]. Then a number of users (students) freely use the WSs under evaluation through a set of clients that support all their exposed functionalities for a pre-specified period of time. During this, the different dynamic quality measures identified in Section IV.B are collected using the data collection framework described in details in Section V of this paper. The framework collects the required data and automatically calculates the average values for each quality measure.

The sub-quality factors *Availability*, *Accessibility* and *Successability* are calculated using the three formulas presented in Section II.A. The data required for calculating *Availability* are collected from the log information of the WSs application server. This includes server's up-times and any possible down-time. The data required for calculating *Accessibility* and *Successability* are collected by capturing the HTTP messages exchanged between the WSs application server and the clients. This allows for calculating the number of request, response, and acknowledgment messages exchanged between the WSs and their clients.

D. Data analysis

After collecting the necessary data points, we will then use statistical regression analysis to identify possible correlation between the software qualities described above for a specific WS and the external software qualities *Availability*, *Accessibility* and *Successability* measured at run-time. We propose logistic regression for our analysis as the values of all the three external qualities (the dependent variables) range from 0 to 1 and the logistic regression curve (Fig. 1) better fit such values.

TABLE I. PRELIMINARY EXPERIMENTAL RESULTS

	Static Measures (average)							**Dynamic Measures (average)**			
	LCO	**CC**	**WMC**	**LCOM**	**I**	**Ca**	**Ce**	**DCBO**	**OMI**	**DC**	**DMI**
Code2web	565	2.26	7.6	0.24	1	0	10	2.00	3.375	1.50	23.00
Artist-Registry	322	1.56	14.2	0.39	1	0	4	1.50	2.09	1.80	14.00

	External Sub-quality Factors		
	Availability	**Successability**	**Accessibility**
Code2web	1	0.998	0.927
Artist-Registry	1	1	0.971

V. THE DATA COLLECTION FRAMEWORK

To achieve the objectives listed in Section III, we designed, implemented and tested a framework for the automatic data collection and metrics calculations. The framework can support developers of WSs in assessing in a simple way the external qualities of their WSs at deploy-time, and to react promptly in case their WSs do not satisfy the expected quality requirements. Server-side, the framework simplifies the process of converting Java applications into WSs, guaranteeing a reliable message exchange between the clients and the WSs. The server-side components are also responsible for the computation of static measures, for creating the environment that is able to compute dynamic measures in a transparent way, and also for calculating *Availability*, *Accessibility* and *Successability* for the target WS.

In the following sections, the framework and its components are described in details.

A. Server-side

The server-side of the measurement framework is centered around the application server Apache Tomcat. First the WSs engine Apache Axis2 is deployed into Tomcat and used to expose (web) applications functionality as standard WSs that communicate using SOAP messages over the HTTP protocol. The targeted WSs are then deployed into Axis2 engine.

To assure reliable message exchange between the clients and the WSs, they were instrumented using Sandesha2 (an implementation of the OASIS WS-ReliableMessages standard [17]). Sandesha2 provides a mechanism that can accurately track and monitor message exchanges between the WSs and their clients. It allows for the accurate determination of the correct disposition of messages only once and therefore, avoid any problems or errors associated with lost or duplicated messages. Using Sandesha2, each request received from the client is acknowledged separately. This facilitates the calculation of the *Accessibility* since it is calculated as the number of acknowledge messages received by the client divided by the number of request messages sent.

Static measures defined in Section IV.B are calculated before the deployment of the WSs into Tomcat using CodePro AnalytiX and the Eclipse Metrics plugin.

Conversely, the dynamic measures defined in Section IV.B are collected using the Aspect-Oriented Programming (AOP) technology [18] at run-time. Each measure is implemented as an "Aspect" that is constructed of "point cuts" and "advices." The "point cuts" define the points in the program runtime flow that are of interest. For example, "point cuts" can be placed to identify each "method call" in the program flow. "Advices" are used to collect data at the defined "point cuts" and to use the collected data to calculated a specific measure. By placing "point cuts" at "method calls," an advice can be used for example, to collect the data necessary to calculate the number of invocation of each method in the program. All dynamic metrics defined in Section IV.B are implemented in a similar way according to their definitions and weaved into the services code during compilation. The generated byte-code is then deployed into Tomcat. When a WS is invoked during a runtime session, the weaved aspects collect all the defined dynamic measures and store the output as text files on the server-side.

During WS invocations, message exchanges between the WS and its clients are captured using the network transport capturing tool WinPcap [19] that captures outgoing and incoming TCP packets to the WS server machine. Wireshark [20] is a network protocol analyzer that is used after each predefined capturing session to (1) extract all HTTP communications, and (2) calculate the number of request, response and acknowledge messages. These data are used to calculate the *Availability*, *Accessibility* and *Successability* of the WS.

B. Client-side

WSs clients are simple Java applications that invoke the WSs under test to deliver its specified functionality. For each WS, a web client is developed and used (or planned to be used) by users in experimental setup to stimulate the WSs while collecting the data necessary to calculate the targeted measures of the WSs. All develop clients for the WSs under evaluation are instrumented by Sandesha2 to support reliable messaging.

VI. Preliminary results

Before executing the planned controlled experiment, we carried out some tests to validate the data collection framework we described in Section V. For this purpose, WSs clients were developed to emulate intensive use of the WSs under evaluation by randomly generating requests in a randomly generated time intervals (range from 0.5 to 2 seconds). Each of the code2web WS and the Artist-Registry WS were separately tested continuously for a period of 30 minutes using separate clients and the defined quality measures were calculated. The results reported in Table 1 were achieved for each of the code2web WS and the Artist-Registry WS. The reported values of *Availability*, *Accessibility* and *Successability* are either 1 or very close to it. This is due to the fact that these qualities usually require longer measurement periods (weeks or months) for failures to occur and hence to produce values different than 1. To overcome this obstacle, we are planning to inject random faults.

The outcomes of this study may be affected by two issues (1) using random fault injection to enforce failures, and (2) controlled experiments may result in restrictively generalizable outcomes. Moreover, the population (Web Services) selected for the experiment are all open-source application with maturity level "Production" or "Stable". Therefore, we consider the population representative enough and allows for the generalization of the results. Taking the above mentioned concerns into account, the results of this study may be considered generalizable.

VII. Conclusions and Future Work

In this paper, we presented our ongoing work towards answering the two research questions: (1) Is it possible to build statistically significant probabilistic models for the WSs sub-quality factors *Availability*, *Accessibility* and *Successability*? and (2) How accurate are these models?

Building probabilistic prediction models for the WSs sub-quality factors *Availability*, *Accessibility* and *Successability* has a strong theoretical basis but experimentation is necessary to build and empirically evaluate the accuracy of the models. The framework presented in this paper and the preliminary experimentation on two case studies showed that it is feasible to collect all the data points necessary for the regression analysis to establish possible correlations between the static and dynamic measures identified and the sub-quality factors *Availability*, *Accessibility* and *Successability* of WSs. Based on that, accurate probabilistic models for the mentioned factors may be built.

Our next steps are (1) to identify and deploy additional WSs so that the total number of WSs will be around 20. This will provide sufficient data for (2) performing the planned regression analysis and allows for (3) building more accurate probabilistic models.

References

[1] OASIS Web Services Quality Model (WSQM) Technical Committee, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsqm [retrieved: August, 2013]

[2] OASIS (Organization for the Advancement of Structured Information Standards), https://www.oasis-open.org [retrieved: August, 2013]

[3] Web Services Quality Factors Version 1.0. 22 July 2011. OASIS Committee Specification 01. http://docs.oasis-open.org/wsqm/WS-Quality-Factors/v1.0/cs01/WS-Quality-Factors-v1.0-cs01.html [retrieved: August, 2013]

[4] ISO/IEC 25000, Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE, ISO (2005).

[5] D. Ivanović, M. Carro, and M. Hermenegildo, "Constraint-based runtime prediction of SLA violations in service orchestrations," In Service-Oriented Computing, Springer Berlin Heidelberg, 2011, pp. 62-76.

[6] F. Xing, P. Guo, and M. R. Lyu. "A novel method for early software quality prediction based on support vector machine," In Software Reliability Engineering, 2005. ISSRE 2005. 16th IEEE International Symposium on, IEEE, 2005.

[7] S. Morasca, "A probability-based approach for measuring external attributes of software artifacts," Proceedings of the 3rd International Symposium on Empirical Software Engineering and Measurement, IEEE, 2009, pp. 44-55.

[8] F. Harrell, "Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis," Springer, 2001.

[9] The Artists Registry Web Service, http://yesso.eu/samples/artist-registry.zip [retrieved: August, 2013]

[10] Apache Axis2 (Java), http://axis.apache.org/axis2/java/core/ [retrieved: August, 2013]

[11] Code2Web Toolkit, http://sourceforge.net/projects/code2web [retrieved: August, 2013]

[12] Chidamber, Shyam R., and Chris F. Kemerer. "A metrics suite for object oriented design," IEEE Transactions on Software Engineering, vol. 20, no. 6, 1994, pp. 476-493.

[13] Chidamber & Kemerer object-oriented metrics suite, http://www.aivosto.com/project/help/pm-oo-ck.html [retrieved: August, 2013]

[14] L. Lavazza, S. Morasca, D. Taibi, and D. Tosi, "On the definition of dynamic software measures," ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), 2012, IEEE, 2012, pp. 39,48.

[15] CodePro AnalytiX, https://developers.google.com/java-dev-tools/codepro/doc/ [retrieved: August, 2013]

[16] Eclipse Metrics plugin, http://metrics2.sourceforge.net [retrieved: August, 2013]

[17] OASIS Web Services Reliable Messaging (WS-1 ReliableMessaging) Version 1.1, June 2007, http://docs.oasis-open.org/ws-rx/wsrm/200702/wsrm-1.1-spec-os-01.pdf [retrieved: August, 2013]

[18] G. Kiczales and E. Hilsdale. "Aspect-oriented programming," In ACM SIGSOFT Software Engineering Notes, vol. 26, no. 5, ACM, 2001, doi: 10.1145/503271.503260.

[19] WinPcap, http://www.winpcap.org [retrieved: August, 2013]

[20] Wireshark, http://www.wireshark.org [retrieved: August, 2013]

Measuring Design Quality of Service-Oriented Architectures Based on Web Services

Michael Gebhart
Gebhart Quality Analysis (QA) 82 GmbH
Karlsruhe, Germany
michael.gebhart@qa82.de

Abstract—**For achieving a flexible and maintainable IT, companies increasingly design their IT architecture in a service-oriented manner using web services. As the effectiveness of this transition is influenced by the design of the architecture, patterns and best-practices have been evolved that are expected to be considered during the development process. However, reviewing the architecture regarding these guidelines is complex and time-consuming as a lot of interpretation and calculation has to be performed. This article introduces an approach for efficiently measuring design quality with a focus on the service layer, thus the service interface and service component design. To illustrate the approach, services of an automotive scenario are developed using a product that integrates the introduced concepts.**

Keywords-soa; web service; design; quality; metrics

I. INTRODUCTION

The ability to realize new business requirements within shortest time has become a critical success factor for companies. This requires the IT to be both flexible and maintainable, which constitute main drivers for service-oriented architecture (SOA) projects [1][2]. While SOA does not dictate any technology usage, in most cases web services are applied as their standardization increases the flexibility and maintainability of the architecture from a technical point of view [3]. In this case, the web services are described using the World Wide Web Consortium (W3C) standards Web Services Description Language (WSDL) [4] and XML Schema Definition (XSD) [5]. Furthermore, in some projects the Service Component Architecture (SCA) [6] standardized by the Organization for the Advancement of Structured Information Standards (OASIS) is applied to describe the component model.

In the past, many projects have shown that the success of SOA projects is influenced by the design of the architecture especially its service layer [7]. On a service layer the architecture targets the design of service interfaces, service components, and their dependencies. Decisions, such as the grouping of operations to services and their granularity, impact the achievement of the previously described goals. For that reason in literature many best-practices and patterns have been identified that describe how to design the service layer. Furthermore, companies also establish standards or design guidelines that represent internal experiences and might be company-, industry-, or project-specific. Developers are expected to consider these guidelines during their work. This requires a solid understanding of the guidelines and discipline to not overlook any application. From a project management perspective it is also necessary to ensure a consistent application of the guidelines.

In both cases, the review of developed web services regarding these requirements is complex and time-consuming. Besides the necessary interpretation and solid understanding a manual analysis of every web service and its relations to other services has to be performed. Furthermore, every change requires a new analysis not only of the changed service but – due to interdependencies – of all web services. The necessary effort is costly and mostly cannot be asserted. In addition, with increasing complexity of the architecture measure mistakes become more likely due to the high number of performed calculations. The result is that quality analyses regarding guidelines are often neglected even though they are relevant for the creation of a flexible and maintainable architecture and the success of SOA projects.

This article introduces an approach to simplify those analyses on a service layer by means of appropriate automation or at least semi-automation. For that purpose, existing best-practices and patterns for service interfaces and service components are formalized so that no interpretation effort is necessary and their compliance can be automatically or at least semi-automatically verified. Even though the internal behavior of a service component, such as its implementation using object-oriented languages, influences the quality of the architecture as a whole, in this article the focus is on the service part represented by the service layer. When designing a service-oriented architecture from a strategic point of view, this is the first essential design task that has to be performed. Previous work in the context of service design metrics will serve as basis for this article. In [8], Gebhart et al. introduced metrics for service designs based on the Service oriented architecture Modeling Language (SoaML) that represent design guidelines. These metrics have been demonstrated by a case study in [9]. Combined with work that describes the relation between SoaML and web services [10] service design metrics based on SoaML are transferred to web services based on WSDL, XSD, and SCA. As result, web services can be automatically analyzed regarding wide-spread guidelines. Furthermore, the methodology can be applied on any other company-, industry-, or project-specific design guidelines.

The concept is illustrated using a scenario in the context of automotive manufacturing. In this case, the usage of formalized guidelines helps to systematically design web services and to coordinate several developers. Furthermore, the concepts are integrated into the QA82 Analyzer as product for analyzing software and data. The product enables the automatic measurement of the design quality of the created SOA, thus increases the efficiency.

The article is organized as follows: Section II introduces existing guidelines for web services and their formalizations. The scenario is introduced in Section III. In Section IV, the services are designed using the formalized guidelines and our product. Section V concludes this article and introduces future research work.

II. Background

This section describes guidelines for the design of services in service-oriented architectures that will be considered within the scenario. Furthermore, this work is examined regarding the possibility to be efficiently measured using tool support. The technologies of web services, such as WSDL, XSD, and SCA are not further introduced in this article. They are assumed to be well known.

The service design phase is an essential ingredient of software service engineering that can be defined as the “discipline for development and maintenance of SOA-enabled applications” [11]. The service design phase includes decisions about the interface of a certain service, such as its grouping of operations, and its internal behavior. As services constitute the building blocks of an SOA, they determine its design. For services several best-practices and patterns have been evolved as guidelines.

In [7] and [12], Erl describes numerous patterns for services in particular web services. They have been derived from experiences in real-world projects and provide valuable hints for architects and developers. Nevertheless, all guidelines are only textually describes. This results in ambiguities and requires interpretation before using it in concrete projects. This again may result in faulty applications.

Similar to Erl, also Cohen [13] and Josuttis [14] focus on patterns from a similar point of view. While the guidelines are clearly motivated, their usage in projects requires interpretation. Furthermore, due to the textual description concrete artifacts cannot be checked against these guidelines without manual effort.

A more academic approach is chosen in [15] and [16]. Perepletchikov et al. introduce metrics for quality attributes, such as loose couplings. These metrics consider formalized service designs independent from concrete technologies. The essential benefit of this work is its ability to perform an automatic measurement. However, the motivation of the introduced metrics is not obvious. Work as introduced by Erl and Josuttis is not reflected by the metrics. This is even not possible as Perepletchikov et al. consider an abstract formalization of services. Most of the aspects described by best-practices refer to elements that are not part of this formalization.

Similarly to Perepletchikov et al. also Hirzalla et al. [17] and Choi et al. [18] introduce metrics for services. Also in this work, the metrics are very abstract and cannot be directly applied in projects. They do not represent best-practices as introduced by Erl and Josuttis.

To fill this gap, in previous work we created a quality model that combines best-practices as introduced by Erl et al. with a formalization as used by Perepletchikov et al. [8]. The quality model was aligned with the Service oriented architecture Modeling Language (SoaML) [19] as profile for the Unified Modeling Language (UML) [20] that is meant to replace proprietary UML profiles for services, such as the one developed by IBM [21][22][23]. As result of this work, an SOA formalized using SoaML can be checked against wide-spread guidelines. The usage of SoaML is explained in [24][25] and a case study that applies the metrics is presented in [9]. However, in most cases web services are created or are already existent without a formalization based on SoaML. Furthermore, some guidelines refer to elements that are not part of a SoaML-based description. Thus, an approach is necessary that is applicable on web services directly.

In [10], it is shown how service designs based on SoaML can be transformed into web services using the WSDL, XSD, and SCA. This work was not necessarily created with quality analysis in mind. However, it can be applied to transfer the service design metrics based on SoaML to web services.

The summary of existing work shows, that a lot of good work exists that focuses either on the description of best-practices, patterns, design guidelines etc. for web services or on a formalization of academic metrics. Whilst the former are too abstract to be efficiently measured, the latter are too academic to be comprehensible understandable and motivated. For that reason we use the metrics introduced in [8] that on the one hand represent best-practices and on the other hand are formalized so that they can be automatically measured. They are transformed so that they can be applied on web services using the mapping rules described in [10].

III. Scenario

To illustrate the quality analysis of a service-oriented architecture design, a scenario from automotive manufacturing is chosen.

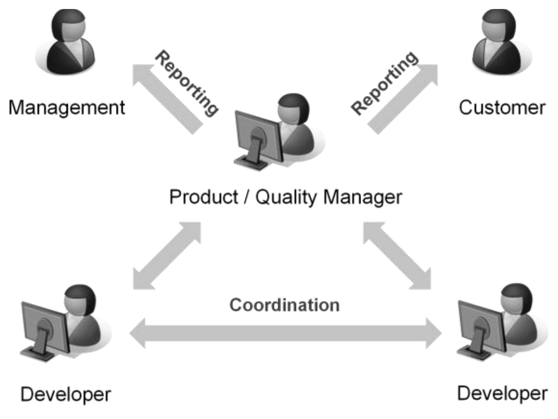


Figure 1. Participants and their relationships.

There is a product and quality manager who coordinates two developers and in addition delivers reports to the management and the customer. In some cases, the role of the product and quality manager might also be fulfilled by an architect, who is responsible for the design of the architecture and its quality. Fig. 1 illustrates the participants and their relationships.

According to this figure, the product and quality manager has an interest in proving the high quality of the created software. In this scenario, besides functional requirements especially the architectural design is considered. So it is necessary that he understands the meaning of high quality in the context of service-oriented architecture design. Furthermore, he is required to analyze software artifacts regarding these quality requirements. To support this quality assurance, this article shows how to analyze artifacts, such as web service interfaces, regarding wide-spread best-practices and guidelines for services.

The scenario begins with the development of a service for the manufacturing of automobiles by the first developer. An SCA Composite is created, which combines a service for manufacturing automobiles and a service for filing manufactured automobiles in the database. The artifacts are filed in a shared Git repository. Fig. 2 illustrates the composite using the graphical representation introduced in the official SCA standard. In the scenario, originally a proprietary tool is used that uses a different visualization.

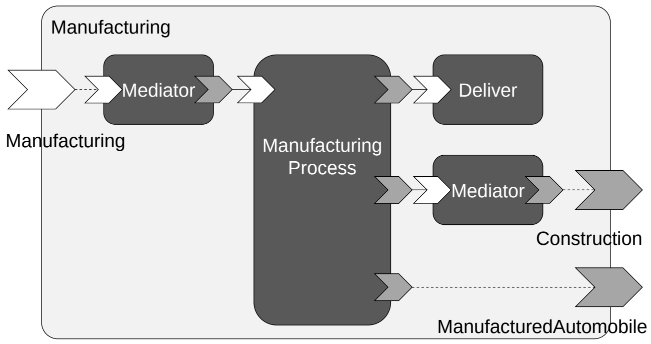


Figure 2. Created SCA composite.

Starting with this SCA composite the product and quality manager determines the quality of the architecture using the approach introduced in the following section.

IV. MEASURING DESIGN QUALITY OF SERVICE-ORIENTED ARCHITECTURES BASED ON WEB SERVICES

To determine the quality of software, one approach is to refine the term quality until it can be measured. A wide-spread quality model methodology is Factor, Criteria, Metric (FCM) introduced by McCall et al. in [26]. According to this methodology a factor is refined into more fine-grained criteria that again are refined into quantifiable metrics. Similar approaches use the equivalent terms quality characteristics, quality sub-characteristics, and quality indicators.

Correspondingly, applied on the design of service-oriented architectures the term quality from a design perspective has to be broken down into measurable aspects that can be formalized by means of metrics. In [8], a quality model has been created that enables the measurement or at least systematic determination of best-practices and patterns that have been identified as important for service-oriented architectures. However, the quality model has been formalized on basis of Service oriented architecture Modeling Language (SoaML) as language to formalize the architecture. When the product and quality manager of the scenario in Section III tries to apply this quality model, the usage of SoaML hampers the direct. As in the scenario other technologies in particular WSDL, XSD, and SCA are used, the metrics introduced in [8] cannot be applied without additional effort. However, in [10], a mapping between SoaML and web service technologies is described. The combination of this work enables the transformation of metrics onto web services so that they can be directly applied. This application is shown next.

A. Application of Metrics

According to Gebhart et al. [8] in particularly four quality sub-characteristics or criteria can be considered as relevant for the design quality: Unique categorization, loose coupling, discoverability, and autonomy. Even though this set of quality characteristics is not expected to be complete it is a good starting point to evaluate the design of a service-oriented architecture and to illustrate the approach.

In this section, especially the unique categorization as quality sub-characteristic is considered. This sub-characteristic is comparable to the concept of cohesion in object-oriented systems. It consists of four quality indicators with metrics introduced in [8][27][28]. To illustrate the approach, these metrics are mapped and applied to analyze the service-oriented architecture design.

1) *Division of Agnostic and Non-Agnostic Functionality:*

TABLE I. VARIABLES AND FUNCTIONS USED FOR DANF

Element	Description and Mapping
DANF	Division of Agnostic and Non-agnostic Functionality
s	service: the considered service that is provided or required It is represented by a SCA Service or Reference element.
SI(s)	Service Interface: service interface of the service s It is represented by the WSDL document that describes the SCA Service or Reference.
RI(si)	Realized Interfaces: realized interfaces of the service interface si. It is represented by the WSDL PortType that includes provided operations of the service.
O(i)	Operations: operations within the interface i The WSDL Operations within the identified WSDL PortType are expected to be returned.
AF(o)	Agnostic Functionality: operations providing agnostic functionality out of the set of operations o This information has to be determined by an IT expert. It cannot be found within the web service technologies.
| o |	Number of operations o

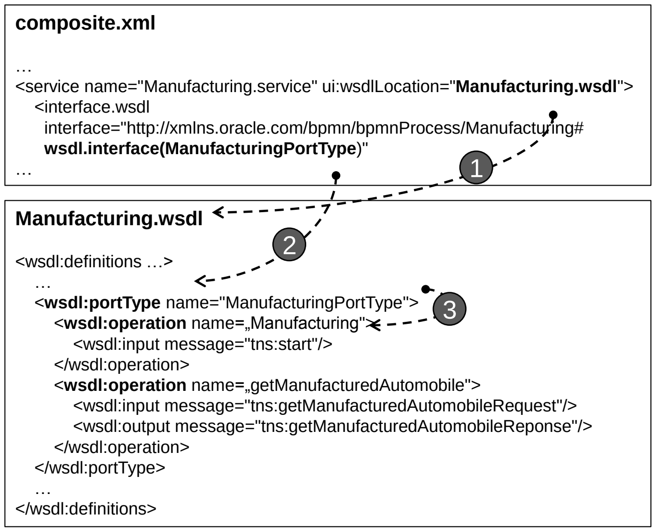
The background of this metric is that generic functionality should be split from specific ones so that changes regarding the specific operations do not affect the highly reused ones. It has its origin in the patterns described by Erl [7].

$$DANF(s) = \frac{\left|AF\Big(O\big(RI(SI(s))\big)\Big)\right|}{\left|O\big(RI(SI(s))\big)\right|} \qquad (1)$$

To apply this metric for the scenario, the functions and variables have to be mapped onto elements within XSD, WSDL, and SCA. Table I shows a brief introduction of the element and afterwards a mapping. This mapping specifies where to find this information.

As result a value of 0 or 1 is desired. These values mean that the service operations provide only agnostic or only non-agnostic functionality.

Based on this mapping information, the metric can be applied for the Manufacturing service that is the SCA Service within the SCA Composite. According to the metric, in a first step the service interface has to be identified. This is the WSDL file Manufacturing.wsdl. Next, the WSDL PortType comprising the provided operations within the WSDL is selected and finally, the operations themselves are returned. Fig. 3 shows the proceeding.



composite.xml

```
…
<service name="Manufacturing.service" ui:wsdlLocation="Manufacturing.wsdl">
   <interface.wsdl
     interface="http://xmlns.oracle.com/bpmn/bpmnProcess/Manufacturing#
     wsdl.interface(ManufacturingPortType)"
…
```

Manufacturing.wsdl

```
<wsdl:definitions …>
   …
   <wsdl:portType name="ManufacturingPortType">
      <wsdl:operation name=„Manufacturing">
         <wsdl:input message="tns:start"/>
      </wsdl:operation>
      <wsdl:operation name=„getManufacturedAutomobile">
         <wsdl:input message="tns:getManufacturedAutomobileRequest"/>
         <wsdl:output message="tns:getManufacturedAutomobileReponse"/>
      </wsdl:operation>
   </wsdl:portType>
   …
</wsdl:definitions>
```

Figure 3. Determination of DANF metric.

After the relevant operations have been identified, the IT quality manager has to decide whether these operations are agnostic or non-agnostic. If he is not capable to answer these questions, he has to ask the developers and estimate the reusability of these operations. In this case, the quality manager comes to the conclusion that the operation "Manufacture" is non-agnostic as it is very specific and cannot be used in other contexts. The operation "getManufacturedAutomobiles" however is agnostic as it provides functionality to request manufactured automobiles, which can be reused in several scenarios. As result the metric returns 0.5, which represents a suboptimal value.

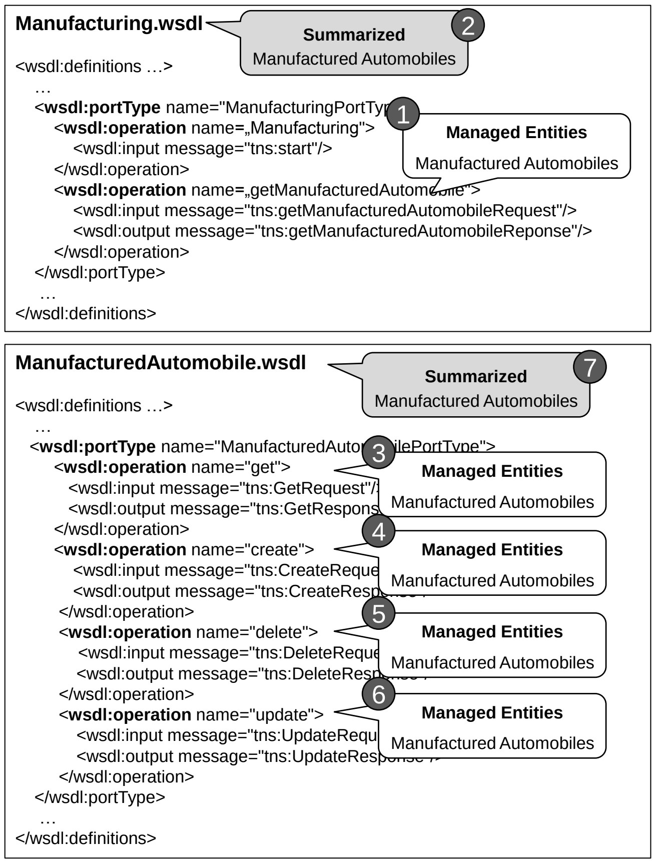
2) Division of Business-Related and Technical Functionality: A metric similar to DANF is DBTF that targets the division of business and technical functionality. It can be mapped in a similar way. To illustrate the approach a more complex metric, the data superiority, is chosen next.

3) Data Superiority: This quality sub-characteristic describes that a service that manages an entity is exclusively responsible for managing it. The metric can be formalized as follows. Most functions have already been described. The others are explained in Table II.

$$DS(s) = 1 - \frac{\left|\begin{matrix} ME\Big(O\big(RI(SI(s))\big)\Big)\cap \\ ME\Big(O\Big(RI\big(SI((ALL_S \setminus s))\big)\Big)\Big)\end{matrix}\right|}{\left|ME\Big(O\big(RI(SI(s))\big)\Big)\right|} \qquad (2)$$

TABLE II. VARIABLES AND FUNCTIONS USED FOR DS

Element	Description and Mapping
DS	Data Superiority
M1 \ M2	Elements of set M1 without elements of set M2 or the element M2
ALL_S	All existing services Represented by all SCA Services
ME(o)	Managed Entities: entities that are managed by operations o This information has to be determined by an IT expert. It cannot be found within the web service technologies.



Manufacturing.wsdl

```
<wsdl:definitions …>
   …
   <wsdl:portType name="ManufacturingPortTyp
      <wsdl:operation name=„Manufacturing">
         <wsdl:input message="tns:start"/>
      </wsdl:operation>
      <wsdl:operation name=„getManufacturedAutom
         <wsdl:input message="tns:getManufacturedAutomobileRequest"/>
         <wsdl:output message="tns:getManufacturedAutomobileReponse"/>
      </wsdl:operation>
   </wsdl:portType>
   …
</wsdl:definitions>
```

ManufacturedAutomobile.wsdl

```
<wsdl:definitions …>
   …
  <wsdl:portType name="ManufacturedAuto
     <wsdl:operation name="get">
        <wsdl:input message="tns:GetRequest"/
        <wsdl:output message="tns:GetRespons
     </wsdl:operation>
     <wsdl:operation name="create">
        <wsdl:input message="tns:CreateReque
        <wsdl:output message="tns:CreateResp
      </wsdl:operation>
      <wsdl:operation name="delete">
         <wsdl:input message="tns:DeleteReque
         <wsdl:output message="tns:DeleteResp
      </wsdl:operation>
      <wsdl:operation name="update">
         <wsdl:input message="tns:UpdateRequ
         <wsdl:output message="tns:UpdateResp
      </wsdl:operation>
   </wsdl:portType>
   …
</wsdl:definitions>
```

Figure 4. Determination of DS metric.

To illustrate this metric we assume that the ManufacturedAutomobile Reference within the SCA Composite refers to a service described by the ManufacturedAutomobile.wsdl and that no other services are relevant for this metric.

To calculate the metric, the product and quality manager has to consider the provided operations of the Manufacturing service and of all other services, i.e., the ManufacturedAutomobile service in this case. Afterwards, he has to decide for each operation whether an entity is managed by this one. Finally, he has to compare the set of managed entities of the services to identify conflicts. Fig. 4 illustrates the proceeding for the Manufacturing service. According to this figure all entities managed by the Manufacturing service are not exclusively managed. The Manufactured Automobile service that corresponds to an entity service [1][7] manages manufactured automobiles too. So from a data superiority perspective the Manufacturing service is not ideal and should be revised.

4) Common Entity Usage: Finally the last quality indicator of the unique categorization quality sub-characteristic can be measured. According to the common entity usage metric, all operations within a service should work on the same entities. This guarantees that entities that do not belong together are managed by different services. In turn, the prior described data superiority ensures that operations that manage the same entities are part of one service.

$$CEU(s) = \frac{\left| OUE\left(\begin{matrix} O\Big(RI\big(SI(s)\big)\Big), \\ CMP\left(\begin{matrix} O\Big(RI\big(SI(s)\big)\Big), MOUE\Big(O\Big(RI\big(SI(s)\big)\Big)\Big), \\ UE\Big(O\Big(RI\big(SI(s)\big)\Big)\Big) \end{matrix} \right) \end{matrix} \right) \right|}{\left| O\Big(RI\big(SI(s)\big)\Big) \right|} \tag{3}$$

TABLE III. VARIABLES AND FUNCTIONS USED FOR CEU

Element	Description and Mapping
CEU	Common Entity Usage
CMP(o, e1, e2)	Composition: biggest set of entities managed by operations o out of e2 that depend on entitites e1
UE(o)	Used Entities: entities that are used within operations o as input
MOUE(o)	Mostly Often Used Entities: entities that are mostly often used within one operation out of operations o
OUE(o, be)	Operations Using Entities: operations out of operations o that only use entities out of be

This table shows that there is no explicit mapping to web services necessary. All functions that refer to certain elements within a technology have already been mapped by the functions described in Table I and Table II.

Applied on the Manufacturing service the metric returns the value 1 as all operations that manage entities manage the same. This is also the case for the Manufactured Automobile service. As this entity service provides Create, Read, Update, Delete (CRUD) operations for the same entity, this metric is also ideal for this service. If the Manufactured Automobile service would also manage another entity, the CEU metric would return a suboptimal value.

B. Integration into Scenario

Back in our scenario, the quality manager can use the results to inform developers about the design weaknesses. The usage of these metrics in a quality-oriented service design process is illustrated in [29].

For example, the result of DANF shows that the two provided service operations “Manufacture” and “getManufacturedAutomobiles” should be separated into two services. In addition, the result of the DS metric shows the conflict between the operations provided by the Manufactured Automobile service and the operation “getManufacturedAutomobile” of the Manufacturing service. Summarized, the operation “getManufacturedAutomobile” should be deleted as it provides functionality that is also offered by the Manufactured Automobile service. Service consumers using this operation should switch to the Manufactured Automobile Service.

In addition to the revision hints, the results of the metrics can be used to deliver reports to the management and the customer. For example the product and quality manager can justify cost and investments into quality assurances. Furthermore, he can prove the quality of the software by means of objective criteria.

V. CONCLUSION AND OUTLOOK

In this article, an approach was illustrated to measure the design quality of service-oriented architectures regarding wide-spread best-practices and guidelines. For that purpose an existing quality model that refers to SoaML as formalization of a service-oriented architecture design was chosen. By use of another work that describes the mapping between SoaML and web service technologies, this quality model was transferred onto WSDL, XSD, and SCA. By this means the resulting quality model can be directly applied on service-oriented architectures based on web services. The approach demonstrated that for an efficient quality assurance existing quality models should be mapped onto the used technologies.

After an examination of existing work, a scenario from automotive manufacturing was introduced. In this scenario, a product and quality manager is responsible to ensure the quality of the resulting architecture. Next, the mapped quality model was applied to measure the design quality of services in this scenario. The metrics mapped onto web services enable the product and quality manager to identify weaknesses in the current design and thus give the developers hints about possible improvements. In addition, the results can be used to deliver reports to the management and the customer. The reports help to prove the high quality and to justify investments in additional quality assurance projects. Furthermore, developers can perform analyses by themselves. The metrics reduce the additional effort to interpret the textual descriptions. Furthermore, they directly refer to concrete elements within the used technologies.

As part of our research work, we have created a mapping for all metrics introduced in [8]. We also implemented this quality model as part of the QA82 Analyzer [30]. Through this both product and quality managers and developers can automatically measure their service-oriented architecture regarding the quality model. This further increases the efficiency of the quality assurance process.

For the future, we plan to include further quality characteristics both regarding service-oriented architectures and related fields. First, we plan to adapt the approach to analyze services based on REST as it is often applied today. As REST does not prescribe certain interface formalization, we assume that the adaptation will require using more implementation-specific information. Second, we work on a quality model for business process management (BPM) that enables the determination of quality characteristics regarding the functional quality of modeled business processes based on the Business Process Model and Notation (BPMN) 2.0 [31]. This quality model is expected to be linked with the experiences we gained with the quality model introduced in this article. The results of this BPM quality model will be published as well. Furthermore, it will be supported by our quality analysis product. Finally, we aim to formalize the described metrics in a technology-independent but executable way. With languages, such as OCL [32] or XQuery [33] it is possible to describe queries that refer to a certain technology, such as UML or XML. We will examine the applicability of these languages for our purposes.

REFERENCES

[1] T. Erl, Service-Oriented Architecture – Concepts, Technology, and Design, Pearson Education, 2006. ISBN 0-13-185858-0.

[2] D. Krafzig, K. Banke, and D. Slama, Enterprise SOA – Service-Oriented Architecture Best Practices, 2005. ISBN 0-13-146575-9.

[3] T. Erl, Web Service Contract Design & Versioning for SOA, Prentice Hall, 2008. ISBN 978-0-13-613517-3.

[4] W3C, “Web Services Description Language (WSDL)”, Version 1.1, 2001.

[5] W3C, “XML Schema Part 0: Primer Second Edition”, 2004.

[6] Open SOA (OSOA), “Service component architecture (SCA), sca assembly model V1.00”, http://osoa.org/download/attachments/35/SCA_AssemblyModel_V100.pdf, 2009. [accessed: January 04, 2011]

[7] T. Erl, SOA – Principles of Service Design, Prentice Hall, 2008. ISBN 978-0-13-234482-1.

[8] M. Gebhart and S. Abeck, “Metrics for evaluating service designs based on soaml”, International Journal on Advances in Software, 4(1&2), 2011, pp. 61-75.

[9] M. Gebhart and S. Sejdovic, “Quality-oriented design of software services in geographical information systems”, International Journal on Advances in Software, *5*(3&4), 2012, pp. 293-307.

[10] M. Gebhart and J. Bouras, “Mapping between service designs based on soaml and web service implementation artifacts”, Seventh International Conference on Software Engineering Advances (ICSEA 2012), Lisbon, Portugal, November 2012, pp. 260-266.

[11] W. van den Heuvel, O. Zimmermann, F. Leymann, P. Lago, I. Schieferdecker, U. Zdun, and P. Avgeriou, „Software Service Engineering: Tenets and Challenges”, 2009.

[12] T. Erl, SOA – Design Patterns, Prentice Hall, 2008. ISBN 978-0-13-613516-6.

[13] S. Cohen, “Ontology and Taxonomy of Services in a Service-Oriented Architecture”, Microsoft Architecture Journal, 2007.

[14] N. Josuttis, SOA in Practice, O'Reilly Media, 2007. ISBN 978-0-59-652955-0.

[15] M. Perepletchikov, C. Ryan, K. Frampton, and H. Schmidt, “Formalising service-oriented design”, Journal of Software, Volume 3, February 2008.

[16] M. Perepletchikov, C. Ryan, K. Frampton, and Z. Tari, “Coupling metrics for predicting maintainability in service-Oriented design”, Australian Software Engineering Conference (ASWEC 2007), 2007.

[17] M. Hirzalla, J. Cleland-Huang, and A. Arsanjani, “A metrics suite for evaluating flexibility and complexity in service oriented architecture”, ICSOC 2008, 2008.

[18] S. W. Choi and S. D. Kimi, “A quality model for evaluating reusability of services in soa”, 10th IEEE Conference on E-Commerce Technology and the Fifth Conference on Enterprise Computing, E-Commerce and E-Services, 2008.

[19] OMG, “Service oriented architecture modeling language (SoaML) – specification for the uml profile and metamodel for services (UPMS)”, Version 1.1, 2012.

[20] OMG, “Unified modeling language (UML), superstructure”, Version 2.2, 2009.

[21] S. Johnston, “UML 2.0 profile for software services”, IBM Developer Works, http://www.ibm.com/developerworks/rational/library/05/419_soa/, 2005. [accessed: July 11, 2012]

[22] U. Wahli, L. Ackerman, A. Di Bari, G. Hodgkinson, A. Kesterton, L. Olson, and B. Portier, “Building soa solutions using the rational sdp”, IBM Redbook, 2007.

[23] A. Arsanjani, “Service-oriented modeling and architecture – how to identify, specify, and realize services for your soa”, IBM Developer Works, http://www.ibm.com/developerworks/library/ws-soa-design1, 2004. [accessed: July 11, 2012]

[24] J. Amsden, “Modeling with soaml, the service-oriented architecture modeling language – part 1 – service identification”, IBM Developer Works, http://www.ibm.com/developerworks/rational/library/09/modelingwithsoaml-1/index.html, 2010. [accessed: July 11, 2012]

[25] M. Gebhart, “Service Identification and Specification with SoaML”, in Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments, Vol. I, A. D. Ionita, M. Litoiu, and G. Lewis, Eds. 2012. IGI Global. ISBN 978-1-46662488-7.

[26] J. A. McCall, P. K. Richards, and G. F. Walters, “Factors in software quality”, 1977.

[27] M. Gebhart, M. Baumgartner, S. Oehlert, M. Blersch, and S. Abeck, “Evaluation of service designs based on soaml”, Fifth International Conference on Software Engineering Advances (ICSEA 2010), Nice, France, August 2010, pp. 7-13.

[28] M. Gebhart, S. Sejdovic, and S. Abeck, “Case study for a quality-oriented service design process”, Sixth Internation Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, October 2011, pp. 92-97.

[29] M. Gebhart and S. Abeck, “Quality-oriented design of services”, International Journal on Advances in Software, 4(1&2), 2011, pp. 144-157.

[30] Gebhart Quality Analysis (QA) 82, QA82 Architecture Analyzer, http://www.qa82.de. [accessed: July 11, 2012]

[31] OMG, “Business process model and notation (BPMN)”, Version 2.0 Beta 1, 2009.

[32] Object Management Group, “Object constraint language”, Version 2.0, 2006.

[33] W3C, “XQuery 1.0: an XML query language (second edition)”, Version 1.0, 2010.

Towards Automatic Performance Modelling Using the GENERICA Component Model

Nabila Salmi
MOVEP Laboratory, USTHB
Algiers, Algeria,
LISTIC Laboratory, Université de Savoie
Annecy le Vieux, France
Email: nsalmi@usthb.dz

Malika Ioualalen
MOVEP laboratory
USTHB University
Algiers, Algeria
Email: mioualalen@usthb.dz

Mehdi Sliem
MOVEP laboratory
USTHB University
Algiers, Algeria,
Email: msliem@usthb.dz

Abstract—**Software designers are often interested in predicting performances of their designed applications, especially for component-based software design where high quality is targeted. In this context, several technics have been proposed. However, none of these approaches has gained widespread industrial use, and automatic tools supporting component-based systems analysis are needed. In this objective, we propose, in this paper, a novel general component model, called *GENERICA*, enabling the description of component-based systems unifying software and hardware components, as well as their deployment and runtime environments and performance characteristics. The aim of this new model is to help designers in deriving automatically performance models, allowing thus automatic qualitative and quantitative analysis of component-based applications, basing on architecture descriptions and component behaviours. The Architecture Description Language (ADL) of GENERICA combines software and hardware components, and allows to describe component-based configurations with performance annotations. Targeted generated performance models consist of Stochastic Petri Nets (SPN) and Stochastic Well-formed Nets (SWN).**

***Keywords*-Component-Based Systems; software component; hardware component; performance annotations; performance modelling.**

I. Introduction

Component-based design of systems is more and more applied for building modern complex hardware and software systems. In this approach, precompiled elementary components, with explicitly defined provided and required interfaces, are being assembled together [1]. Such systems are known as Component-Based Systems (CBS). The main goals are to improve software quality and to reach reduced cost and easy maintaining and upgrade. Several academic and industrial component models have been developed, such as Fractal [2], EJB [3], CCM [4], AADL [5], Palladio [6], Koala [7], etc.

Very often, designers are interested in predicting performances of their designed systems (such as response times, throughput, etc.), to avoid performance problems after implementation, which can lead to re-designing substantial costs. It would be helpful if the designer can automatically perform an "a priori" analysis of his/her systems. This requires to generate automatic component modelling. As component performance depends not only on implementation, but also on the context the component is deployed in, it would be beneficial if we can get deployment and runtime environment information directly from the system architecture description or from some other component specification tools, to automatically build a performance model for a system. Indeed, we need two information categories: on one side, the runtime environment nature (e.g., hardware components, middleware components, etc.); on the other side, information about context performance (e.g., processor rate, memory space, number of component threads, etc.). Despite the numerous proposed component models, only few of them offer such information or some related specifications, such as Palladio [6] and Procom [8].

On this behalf, we attempt, in this work, to provide a component model with necessary information enabling automatic component performance modelling; we propose a general component model with its Architecture Description Language (ADL) allowing to describe component properties, as well as deployment and runtime environment and performance characteristics. These specifications are used to derive from architecture descriptions and component behaviours an automatic mapping into performance models, without additional modelling. So, we describe here the *GENERICA* model, developed for this aim. Targeted generated performance models are Stochastic Petri Nets (SPN) and Stochastic Well-formed Nets (SWN) [9], which are well-known for their expressiveness and existing performance analysis methods and tools, such as GreatSPN [10].

The paper is organized as follows. Section II discusses related work. Then, Section III presents main requirements of a general component model. We detail, in Section IV, our proposal, the GENERICA component model. Corresponding generated performance models are given in Section V. Section VI introduces the GenTools prototype that we developed to support compilation of architecture descriptions of Generic systems and model generation, and illustrates component modelling with an application example. Section VII concludes the paper.

II. Related work

Over the last decades, several component models have been proposed, They have been applied to a large spectrum of application domains. Several classifications and surveys have been also achieved, attempting to identify key features of component software approaches [11][12][13][14][15]. According to the classification done by Crnkovic et al. [12], two kinds of component models are distinguished: general-purpose and specialized models. General-purpose models have similar solution patterns, whereas specialized ones have specific domain characteristics Hence, many component characteristics are not always included in existing component model, and no complete or generic component model gathers all component features, except UML/MARTE [16], which is a quite generic model capturing a large number of systems, even if it is hardly used because of its complexity.

Besides, providing deployment, performance and runtime properties in a component model is uncommon. These properties are extra-functional and their specification is still not widespread. In this context, resource usage and some other performance properties have been modelled by few models such as component models are Palladio, SaveCCM, ProCom, Pin and CompoNETs, as given in some surveys [12][13].

The Palladio component model (PCM) [6] is a domain-specific component model, designed to enable early performance predictions for component-based business software architectures. For that purpose, deployment environment and resource allocation to components are specified using a proper domain-specific modelling language. Then, PCM models are created using an integrated modelling environment, called PCM-bench, and performance metrics are derived from these models using analytical techniques and simulation.

SaveCCM [17] is also a domain-specific component model designed for embedded control automative applications, targeting to provide predictable vehicular systems. It considers resource usage and analysability of the dependability and real-time properties. Component behaviour modelling is done using timed automata extended with tasks. Analysis is then performed at design time using a model checker.

Procom [8] is a component model for control-intensive distributed embedded systems. An extra-functional component behaviour is described in a dense time state-based hierarchical modelling language. This behaviour consists namely in timing, resource consumption, component allocations, etc. Pin [18] is a simple component technology, used also in Prediction-Enabled Component Technologies (PECT). It supports prediction of average latency in assemblies and in stochastic tasks, and formal verification of temporal safety and liveness. Finally, CompoNETs [19] is a general-purpose component model, based on CCM, where, additionally, the internal behaviour of a software component and intercomponent communication are specified by Petri Nets. A mapping from the constructs of the component models to Petri Nets is defined.

From these model descriptions, we deduce that some component models are domain-specific, missing genericity, and others support some behavioural or performance specifications, requiring sometimes deployers or experts to provide such information on the designed application. We want to provide designers with tools allowing them to perform "a-priori" analysis of their systems or applications, basing on automatic generated performance models and without requiring an expert intervention. To do so, the component dimension (which deals with general component and assembly properties) and the performance behavioural dimension should be gathered in the same model, to enable automatic component performance modelling and analysis at design time. However, no model includes at the same time the two kinds of properties (component and performance), and if such model exists, often an expert performs this modelling task for performance analysis of designed systems.

Hence, we introduce in this paper a general component model, the GENERICA model, which fares better along the two dimensions, inspired from two models: the Fractal model [2] and the AADL model [5]. These two models comprise many generic features, as well as UML/MARTE, which make them interesting to use, however they lack performance characteristics. Our component model is a combination of common component/assembly features, runtime environment and performance features: It includes genericity features of Fractal, and hardware component features from AADL, but also allows designers to describe runtime and performance characteristics, as attributes of software and hardware components. Thus, specific software and hardware systems, such as embedded systems, can be modelled using GENERICA. Moreover, the specification of all these features with performance annotations is useful to derive directly from architecture descriptions and component behaviours an automatic mapping into performance models, without additional effort modelling, and hence useful to conduct an automatic a priori qualitative and performance analysis of designed systems.

III. Requirements for a general component model

As defined by Crnkovic et al. [12], a component model defines standards for properties that individual components must satisfy, and methods for composing components. Component properties are commonly known as being functional properties and extra-functional specifications (like quality of service attributes). These properties are exposed by means of interfaces, whereas composing components includes mechanisms for component interaction. These mechanisms are mainly bindings defining connections between interfaces. Besides, modern applications generally run in a multi-layered environment. An application is deployed on an application server, which, in turn, runs on some virtual machine (e.g., Java virtual machine, .NET, etc.). The virtual machine works on an operating system (OS), which utilizes some hardware resources. Such configurations highlight several factors, which may influence performances of an application and particularly performances of a component-based application:

- Number of execution flows (threads) of software components,
- Processing rates of hardware components,
- Processing rates of operating systems and middleware under which the application is running,
- Amount of space memory required during execution,
- Amount of necessary resources allocated to component, and
- Parallel applications running under the same operating system.

Consequently, we identify the following main elements that should be allowed by a general component model:

- Software components, which may be primitive or composite, and whose exposed interfaces may be of any kind (service invocation interface or event-based interface),
- Component bindings, being synchronous (invocation service) or asynchronous (event-based) connections.
- Hardware components composing the deployment and runtime environment, and
- Deployment and performance component features (service rates, used memory, required resources, threads, etc.).

These requirements have led to the GENERICA model, which gathers all these characteristics.

IV. The GENERICA Component model

Our model is defined around common component concepts that are components, interfaces and interactions. To be generic, we allow the definition of software components, hardware components and system configurations describing a component-based application deployed on a running environment (as it is shown in Figure 1). Finally, performance annotations are added to describe performance properties. To describe component architectures following our model, we defined the GENERICA ADL, based on a textual XML syntax.

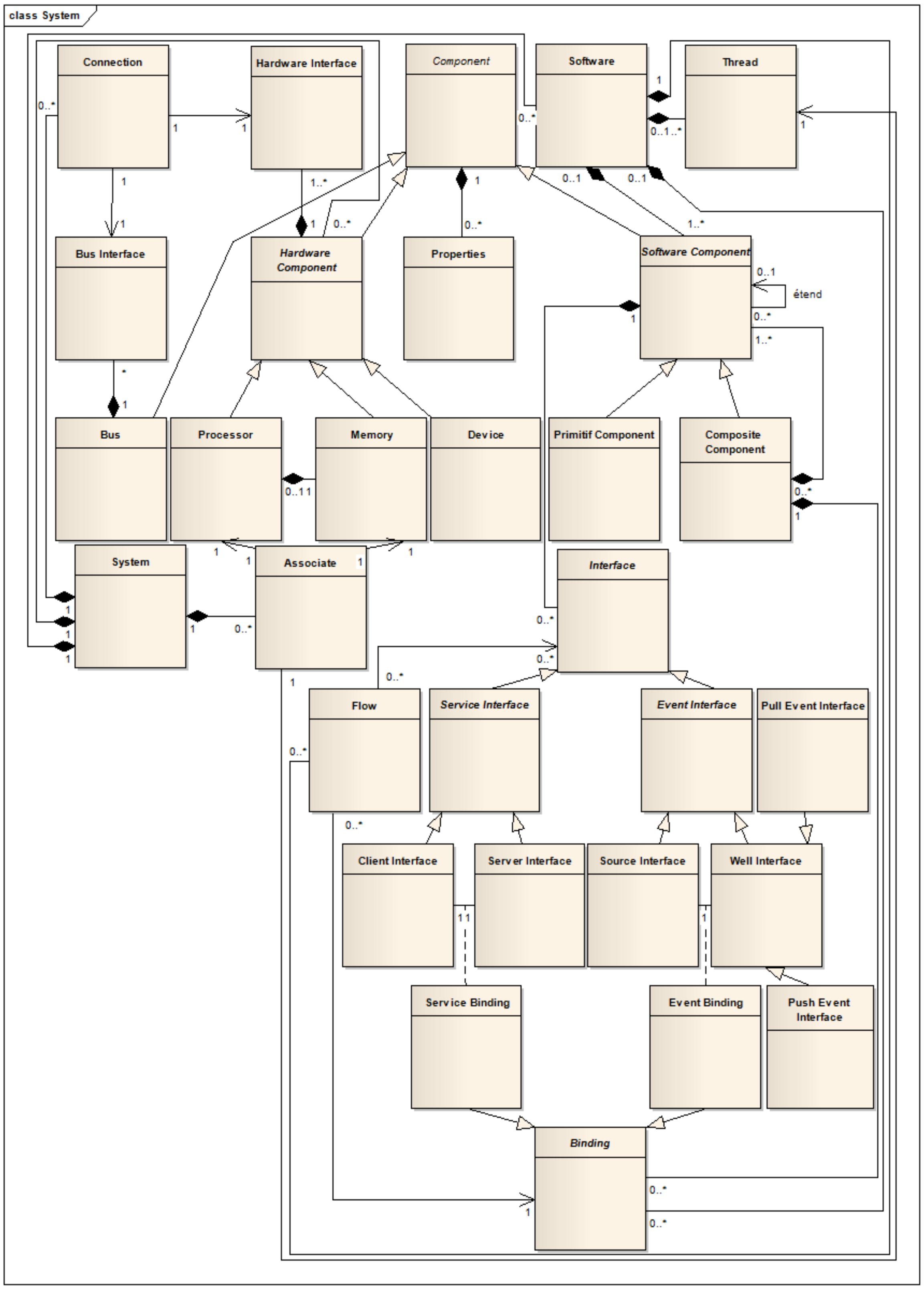


Fig. 2: GENERICA Metamodel

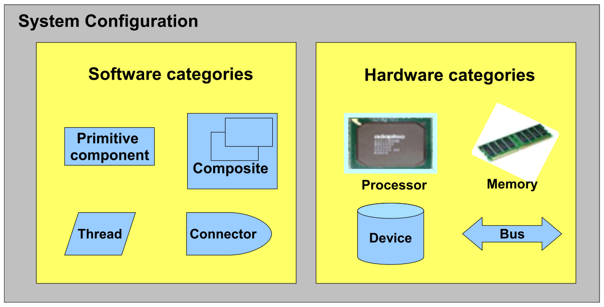


Fig. 1: Software and hardware categories

A. *Software components*

As in other component models, a software component is made of a content and a set of access points, called interfaces, used for interaction with its environment. The content is either composite, composed of a finite number of other components, allowing components to be nested at an arbitrary level, or be primitive (source code) at the lowest level.

An interface can be a functional interface describing component functionalities, or a control interface for non functional properties such as monitoring and control over execution. To be as generic as possible, two sorts of functional interfaces are defined in GENERICA:
• Service invocation interfaces enabling synchronous communications. Two kinds of interfaces are used: a client interface requesting a service, and a server interface providing the service.
• Event based interfaces, resulting in asynchronous communications. In this case, an event source interface generates events and an event sink interface receives event notifications. The reception of a notification causes the acknowledgment of the reception and execution of a specified handler.

To communicate, components are connected by relating their corresponding interfaces through a binding.

Other characteristics can be described by GENERICA: generalization, component inheritance, connectors, sharing, etc.

B. *Hardware components*

The interest in adding hardware components to our model is to allow descriptions of the runtime environment and hardware systems as well. These descriptions enable to do a detailed performance modelling, to be used for qualitative and quantitative system analysis while considering the running platform influences. Four kinds of hardware components are defined in GENERICA:
• Processor: models a processor associated to a minimal OS.
• Memory: represents a storage device.
• Bus: acts for all kind of networks or bus communication.
• Device: defines peripheral or resource elements whose internal structure is ignored. Hardware components interact through bus components, instead of using interfaces.

C. *Threads*

Software components are linked to hardware components by defining component running states being executed on hardware elements. These running states are represented by threads. To describe that in our ADL, one or several threads are first associated to software components; then, these threads are linked to hardware components. At least one thread must be associated to a component application. The main role of thread description is to allow multithreading definition in the generated performance models, so that to enable computation of multithreading impact and performances on the analyzed system.

D. *System configurations*

A GENERICA system configuration consists of a software application mapped to a hardware platform. The mapping is made by describing a semantic connection (or association) between component threads (defined for the software) and hardware components. Consequently, three parts form the system configuration (see the example below):
• A "software" part, where are described software components,
• A "hardware" part, defining hardware components, and
• An "association" part relating component threads to hardware components.
If the designer wants to describe only a software application, it is possible to omit the hardware part description.

E. *Data flows*

Sometimes, when invoking a component service, the called component invokes itself a service from another component, which in turn may call another service, etc. So, data cross several components until executing the first requested service. This case corresponds to a data flow, defined as data routing across the system architecture or dependencies between several requests being service invocations or events notifications. For instance, receiving an event notification on a sink interface of a given component may cause service invocation to another component. Data flows are useful to build a complete detailed knowledge about the studied system, which helps in generating a correct modelling. So, it is important to highlight data flows in an architecture description of a system. For that purpose, GENERICA allows to describe data flows as a dependency between a server interface and a client or source interface of the same component, or between a sink interface and a client or source interface.

F. *Performance annotations*

One of the main contributions of GENERICA is the definition of performance annotations, which will enable later to map components into formal performance models, namely Stochastic Petri Net (SPN) and Stochastic Well-formed Net (SWN) models. For this purpose, we need to specify some information:
• To assess multithreading impact, the number of threads of components and the system configuration is necessary.
• To evaluate service or event processing performances (such as response times or throughput), we need to know the processing rate as well as code size or an estimated execution time of a service or event processing.
• When interest is given to storage size, we need the storage capacity or speed, the data bus speed and dataflow size.

So, we distinguish the following annotations appearing as attributes added to corresponding elements:
• Four annotations for hardware components: data bus speed, processing rate, storage capacity and processor scheduling strategy. Note that this information is useful for assessing software components running even on the same or heterogenous systems with different processor families for instance.
• Four annotations for software components: estimated number

of thread instructions, estimated execution time of a service method, dataflow size, and request arrival rate.

All described core concepts of the GENERICA component model are gathered in its metamodel shown in Figure 2.

V. Modelling with SPN/SWN

From the main characteristics of the GENERICA model, we derive a generic approach for automatic model generation of GENERICA systems, inspired from previous work [20]. The proposed modelling is based on the Stochastic Petri Net (SPN) and Stochastic Well-Formed Petri Net (SWN) models; the SWN model being a high level (coloured) model of Petri Nets with probabilistic extensions for performance analysis [9]. These formalisms are state based models, well known for being able to model complex systems with concurrency and conflicts, and widely used for qualitative and performance analysis. In particular, the SWN model is well suited for behavioral symmetries of system's entities.

Let be a GENERICA system defined through an ADL description and a set of Java classes corresponding to primitive components. To generate a model for this system, we first model each primitive component. For this purpose, as a GENERICA component may be made of a local behaviour (set of internal actions) and a set of interfaces, we defined basic SPN/SWN models for interfaces and internal component behaviour. Using these basic models, each primitive component is modelled. Finally, we generate the GENERICA system global model, using previously generated SPN/SWN models. This global model highlights components and component communication, and hence, bottlenecks can be detected within this model. More details can be found in [20].

VI. Illustration

A first tool helping in building an GENERICA architecture description has been developed: the GENERICA toolbox.

A. The GENERICA tool prototype

The GENERICA component model ADL has been implemented into a Java prototype GenTools, using the Java language. This prototype provides an editor for introducing an ADL system description, a compiler to check syntactical and semantical errors and an SPN/SWN model generator for primitive components and for the whole application. To do a qualitative or/and performance analysis of generated models for a given application, we need to use existing SPN/SWN analysis tools such as the GreatSPN tool [10]. It would be interesting to have such analysis automated after model generation. This is one of our future work. The user interface of the GENERICA toolbox is depicted in Figure 3, showing an application example with its generated model.

B. Running example

To illustrate the description of a component-based application using the GENERICA model, we use a typical industrial application (Figure 4), the stock quoter system, which is an extended version of an application presented in [21]. This application is a system managing a stock information database, chosen mainly for its components exposing, at the same time, service invocation and event-based interfaces. When the values of particular stocks change, a StockDistributor component sends an event message that contains the stock name to two StockBroker components. If the first StockBroker component is interested in the stock, it can obtain more information about it, by invoking a service operation offered by an Executor component. This latter processes the received request, generates data and invokes itself a service request from a persistence server component to save its results. Besides, if the second StockBroker component is interested in the stock, it processes locally the event. Figure 4 shows the interactions between the different components.

```
<system name="Stock Quoter">
 <!-- Hardware architecture -->
 <hardware>
  <components>
    <processor name="processor1">  <interface name="IntP"/> </processor>
    <memory name="memory1">  <interface name="IntM"/> </memory>
     ...
  </components>
  <connections>
     ...
  </connections>
 </hardware>
 <!-- Software architecture -->
 <software name="Stock application">
  <components>
   <component name="StockDistributor1">
     <interfaces> <event name="newst" role="source"/>  </interfaces>
   </component>
   <component name="StockBroker1">
     <interfaces> <event name="rec" role="sink"/>  <service name="rqst" role="client"/>
     </interfaces>
     ...
  </component>
   .. .
  <component name="Executor">
     <interfaces> <service name="prc" role="service"/> <service name="save" role="client"/>
     </interfaces>
     ...
  </component>
   ...
  </components>
  <bindings>
    ...
  </bindings>
  <flows>
   <flow name="fl1" in-component="StockBroker1" for-interface="rqst" for-dependency=
     "execute" take-binding="BrokExec"/>
      ...
  </flows>
  <threads><thread name="thread1" /></threads>
 </software>
 <!--Hardware&Software Associations-->
 <associations>
   <associate name="ass1" processor="processor1" memory="memory1" thread="thread1"/>
 </associations>
</system>
```

Fig. 5: Part of the GENERICA ADL of the application example

Figure 5 shows a part of the architecture description of the application using the GENERICA ADL.

The generated SPN global model is depicted on the user interface of Figure 3.

VII. Conclusion and Future work

This paper presented the GENERICA component model, a new general model, which deals with two dimensions: the component dimension describing general component and assembly characteristics, and the performance behavioural dimension related to deployment, runtime environment and performance properties. An ADL language has been also proposed for GENERICA, as well as a corresponding performance modelling approach based on SPN and SWN models. The long-term objective of introducing such a general component model, is to enable automatic performance component modelling and hence automatic a priori qualitative and performance analysis of component based systems. Even if introduction of a generic component model may lead to a complex specification, we think of its usefulness for several design fields such as embedded systems. This component model has been implemented into a Java toolbox prototype, the GenTools toolbox, supporting compilation of ADL descriptions and model generation. The tool has been experimented on several GENERICA applications.

However, still more research work is required in several directions, such as integrating the GENERICA toolbox in a global modelling and analysis tool, starting from the ADL description and automatic modelling, and resulting in performance computations, given specification of performance indexes of interest. We also target to use the automated modelling of primitive components in a compositional analysis step, based on components models, to have time and memory savings during models analysis. This can be done thanks to our previous work [20], which defined a structured performance analysis method for analysing a CBS in an efficient way allowing, to reduce computation times and memory usage (basing on primitive component models rather than the global net). Finally, we are working on modelling reconfiguration features of GENERICA CBSs and verification of their behaviours.

References

[1] C. Szyperski, *Component software*, 2002, vol. 2nd Edition.

[2] E. Bruneton, T. Coupaye, and J. Stefani, "The fractal component model, version 2.0-3," http://fractal.ow2.org/specification/ (October 2013), Tech. Rep., Feb 2004.

[3] Sun Microsystems, "EJB 3.0 specification," http://www.oracle.com/technetwork/java/index.html, Jul 2007.

[4] Object Management Group, "CORBA component model specification. version 4.0," http://www.omg.org/spec/CCM/4.0/ (October 2013), Apr. 2006.

[5] SAE, "Architecture analysis et design language (aadl)," SAE Standards AS550, Tech. Rep., November 2004.

[6] S. Becker, H. Koziolek, and R. Reussner, "Model-based Performance Prediction with the Palladio Component Model," in *WOSP2007*. Buenos Aires, Argentina: ACM Sigsoft, 2007.

[7] R. van Ommering, F. van der Linden, J. Kramer, and J. Magee, "The Koala component model for consumer electronics software," *IEEE Computer*, vol. 33, no. 3, pp. 78–85, Mar. 2000.

[8] S. Sentilles, A. Vulgarakis, T. Bures, J. Carlson, and I. Crnkovic, "A component model for control-intensive distributed embedded systems," in *CBSE*, 2008, pp. 310–317.

[9] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, "Stochastic well-formed colored nets and symmetric modeling applications," *IEEE Trans. on Comp.*, vol. 42, no. 11, pp. 1343–1360, Nov 1993.

[10] Perf. Eval. Group, "GreatSPN home page: http://www.di.unito.it/~greatspn," Torino, Italy, 2002.

[11] H. Aris and S. S. Salim, "State of component models usage: justifying the need for a component model selection framework," *Int. Arab J. Inf. Technol.*, vol. 8, no. 3, pp. 310–317, 2011.

[12] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. R. Chaudron, "A classification framework for software component models," *IEEE Transactions on Software Engineering*, vol. 37, pp. 593–615, 2011.

[13] J. Feljan, L. Lednicki, J. Maras, A. Petricic, and I. Crnkovic, "Classification and survey of component models," Målardalen University, Technical Report ISSN 1404-3041 ISRN MDH-MRTC-242/2009-1-SE, December 2009.

[14] K.-K. Lau and Z. Wang, "Software component models," *IEEE Trans. on Software Engineering*, vol. 33, no. 10, pp. 709–724, October 2007.

[15] N. Medvidović and R. N. Taylor, "A classification and comparison framework for software architecture description languages," in *IEEE Trans. On Soft. Eng.*, vol. 26, no. 1, 2000, pp. 70–93.

[16] S. Taha, A. Radermacher, S. Gérard, and J.-L. Dekeyser, "Marte: Uml-based hardware design from modelling to simulation," in *FDL*, 2007, pp. 274–279.

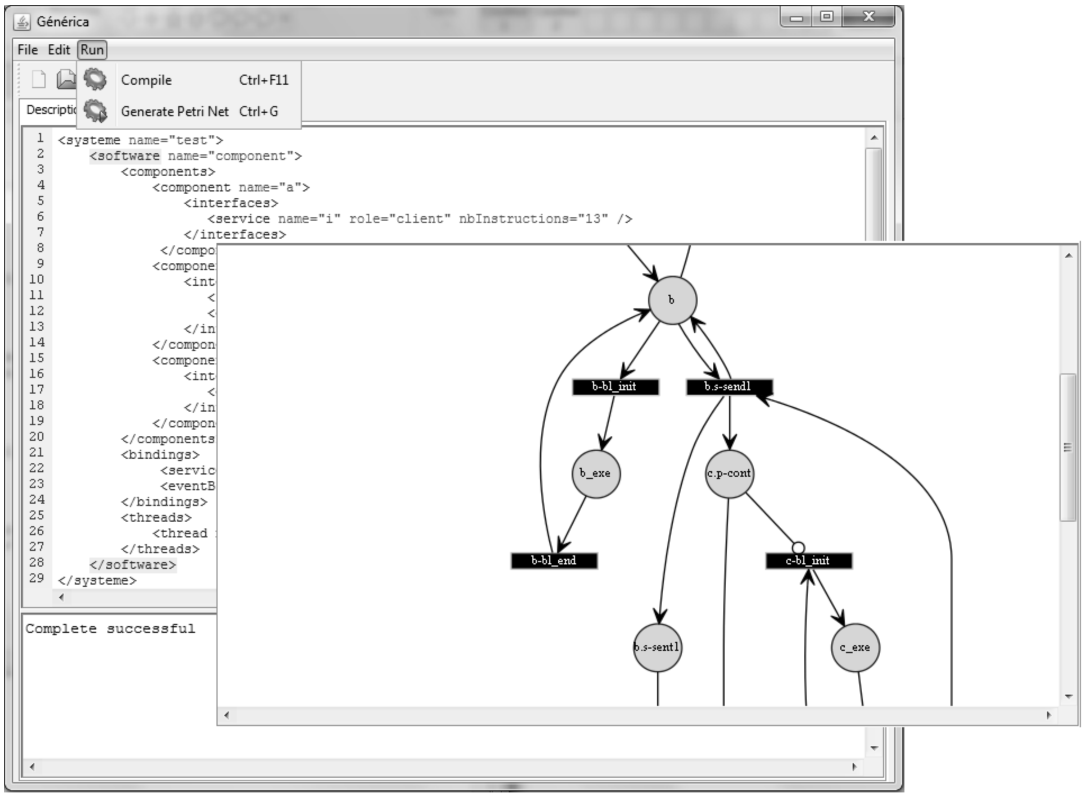
[17] M. kerholm, J. Carlson, J. Fredriksson, H. Hansson, J. Håkansson, A. Möller, P. Pettersson, and M. Tivoli, "The save approach to component-based development of vehicular systems," *J. Syst. Softw.*, vol. 80, no. 5, pp. 655–667, May 2007.

[18] S. Hissam, *Pin Component Technology (V1.0) and Its C Interface*, ser. Technical note. Carnegie Mellon University, 2005.

[19] R. Bastide and E. Barboni, "Component-Based Behavioural Modelling with High-Level Petri Nets," in *MOCA '04- Third Workshop on Modelling of Objects, Components and Agents , Aahrus, Denmark , 11/10/04-13/10/04*. DAIMI, October 2004, pp. 37–46.

[20] N. Salmi, P. Moreaux, and M. Ioualalen, "Structured performance analysis for component based systems," *International Journal of Critical Computer-Based Systems (IJCCBS)- Part II - Issue 1/2*, vol. 3, no. 1, pp. 96–131, 2012.

[21] D. Schmidt and S. Vinoski, "Object interconnections: The CORBA component model: Part 2, defining components with the IDL 3.x types," *C/C++ Users Journal*, April 2004.



```
<systeme name="test">
    <software name="component">
        <components>
            <component name="a">
                <interfaces>
                   <service name="i" role="client" nbInstructions="13" />
                </interfaces>
             </compo
            <compone
                <int
                   <
                   <
                </in
            </compon
            <compone
                <int
                   <
                </in
            </compon
        </components
        <bindings>
             <servic
             <eventB
        </bindings>
        <threads>
            <thread
        </threads>
    </software>
</systeme>
```

```
Complete successful
```

Fig. 3: User interface of the the GenTools tool

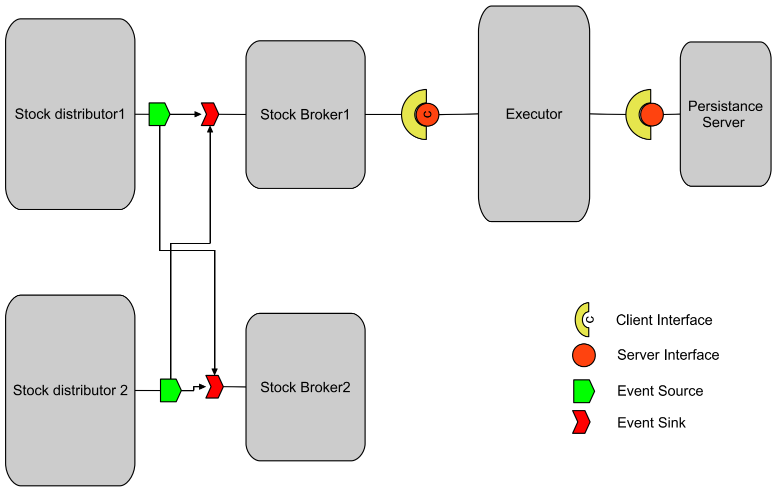


Fig. 4: Application example

Ensuring Consistency of Dynamic Reconfiguration of Component-Based Systems

Hamza Zerguine
MOVEP laboratory
USTHB University
Algiers, Algeria,
Email: hzerguine@usthb.dz

Nabila Salmi
MOVEP Laboratory, USTHB
Algiers, Algeria,
LISTIC Laboratory, Université de Savoie
Annecy le Vieux, France
Email: nsalmi@usthb.dz

Malika Ioualalen
MOVEP laboratory
USTHB University
Algiers, Algeria
Email: mioualalen@usthb.dz

***Abstract*—The introduction of dynamic reconfiguration properties in a system can affect its performance and quality of service offered to users. Thus, performance prediction of component-based systems after reconfiguration is important to help software engineers to analyze their applications at the moment of reconfiguration and take decision to keep or discard the analyzed reconfiguration, so that performance problems are avoided. In this case, the design and verification of functional and non-functional properties before and after reconfiguration become a challenge. In particular, when a applying a reconfiguration on a system, the consistency of the new resulting architecture should be checked. To this aim, we describe, in this paper, a generic reconfiguration analysis approach which allows to check the reconfiguration consistency of a component-based architecture, starting from the architectural description of a component-based system. A case study of a system reconfiguration illustrates the effectiveness of our approach.**

***Keywords*-Component-Based Systems; dynamic reconfiguration; formalization; consistency.**

I. INTRODUCTION

Component-based approaches [1] are more and more essential for the development of systems and applications, to meet the challenges of engineering systems such as administration, autonomy. In this paradigm, components are developed in isolation or reused and are then assembled to build a *Component Based System* (CBS). Their objective is to enable a high degree of reusability of the software, rapid development (reducing the cost in terms of development time) and high quality since development is based on precompiled components In this direction, numerous component models have been proposed (e.g., Enterprise Java Beans (EJB) [2], Corba Component Model (CCM) [3], Fractal [4], etc.). They operate different life-cycle stages, target different technical domains (embedded systems, distributed systems, etc.) and offer different degrees of tool support (textual modeling, graphical modeling, automated performance simulation, etc.).

Nowadays, systems need more and more to adapt their behaviour to their environment changes. To do that, they should dynamically add, remove or recompose components by the use of computational reflection. These abilities are called dynamic or runtime reconfiguration and constitute a key element to enable the adaptation of complex systems, such as embedded systems (mobile phones, PDAs, etc.) and service-oriented systems, to a changing environment. Moreover, dynamic system reconfiguration allows to achieve continuous availability of systems.

Dynamic reconfiguration techniques are promising solutions for building highly adaptable component-based systems. However, the introduction of dynamic reconfiguration properties in a system can affect its performance and quality of service offered to users. To avoid this, the design and verification of functional and non-functional properties of a reconfigured system become a challenge.

In this context, our long-term goal is to develop a methodology which allows analysis of component-based applications and their correction after reconfiguration, to help the decision to keep or discard the analyzed reconfiguration. The first property to ensure during analysis of such systems is consistency, which is defined as remaining compliant with their specification [5]. In this paper, we introduce a new formalism for checking consistency of dynamic reconfigurations of component-based systems. We provide this formalism for general component systems characterized by the most common component properties.

Outline. The structure of the paper is as follows. We discuss in Section II the related work. Then, we present in Section III the most important concepts of component-based systems. We detail our approach in Section IV and illustrate it in Section V. We conclude in Section VI and give future works.

II. RELATED WORK

Several approaches were proposed, during last years, for analysis of CBS; a few of them addressed dynamic reconfiguration.

In this context, two main proposals were given for dynamic reconfiguration analysis of CBS. First, Grassi et al. [6] proposed a metamodel called KLAPER, which includes a kernel modelling language. The main goal of this language is to act as a bridge between design models of component-based systems (built using heterogenous languages like Unified Modeling Language (UML) [7], Ontology Web Language (OWL) (OWL-S) [8], etc.) and performance analysis models (Markov chains [9], queueing networks [10], etc.). This first work did not address reconfiguration cases study. Later, in [11], an extension of KLAPER, called D-KLAPER, was given to support the model-based analysis of reconfigurable component-based systems, with a focus on the assessment of particular non-functional properties, namely performance and reliability.

The second work, defined by Leger [5], targeted dynamic reconfigurations reliability analysis for component-based systems, where an analysis approach for the Fractal component model was defined. The approach is summarized in three steps: the first step is a Fractal configuration modeling [5] step of a component-based configuration architecture; then, definition of mechanisms used for maintaining systems consistency during dynamic reconfigurations; finally, implementation of these mechanisms for checking reliable reconfiguration.

Besides, some other approaches were proposed for CBS for checking, in particular, consistency of CBS during dynamic reconfiguration. Warren et al. [12] proposed to do automatic runtime checks of reconfigurable component-based systems for the OpenRec framework [13]. A formal model based on ALLOY [14] was defined for that purpose. It allows architecture constraints expression and checking. Another work [15] has introduced an extension of the Fractal model [16], called *Safran* to enable the development of adaptive applications. It consists of a dedicated programming language for adaptation policies, as well as a mechanism to dynamically attach or detach policies to or from Fractal basic components. Finally, M. Simonot et al. [17] proposed a formal framework, called *Fracl*, for specifying and reasoning about dynamic reconfiguration programs, being written in a Fractal-like programming style [4]. This framework is based on a first order logic, and allows properties specification and proof concerning either functional or control concerns. An encoding of their model using the Focal specification framework [18] enabled them to prove its coherence and obtain a framework for reasoning on concrete architectures.

These proposals are interesting, however, Safran, Fracl and Leger's proposals are focused on Fractal models only. In particular, Fracl was defined only for applications with primitive components. In addition, no difference is done between *Mandatory* and *Optional* interfaces and no subtyping notion is considered. Warren et al. [12] focused on OpenRec framework only. Moreover, only connections between component are modelled and not component behaviours.

In our case, we target to provide a generic formalism to be used for checking consistency in any component-based system. Our approach formalizes main component elements (component, interfaces, bindings, etc.) and defines for each reconfiguration operation a set of constraints to build consistent configurations. Global constraints are also introduced on a CBS after its reconfiguration.

III. Component Based Systems

A software *component* is defined as a unit of composition, provided with contractually specified *interfaces* and explicit context *dependencies* [19]. An interface is an access point to the component, which defines provided or required services. In addition, types, constraints and semantics are defined by the *component model* in order to describe the expected behaviour at runtime.

Interfaces of a component allow to connect it to other components. Consequently, we build a Component-based System by connecting the interfaces of components. These connections are done depending on interactions between components. Generally, two main styles of interactions are defined in component models: synchronous interactions provided by service invocation (such as an Remote Procedure Call (RPC) or Remote Method invocation (RMI) communication), and asynchronous interactions given through notification of events (asynchronous messages). Service invocations take place between a *client* interface requesting a service and a *server* interface providing the service. Besides, event communications are defined between one or more *event source* interfaces generating events and one or several *event sink* interfaces receiving event notifications. The reception of a notification causes the acknowledgment of the reception and execution of a specified reaction called the *handler* of the event. Some event services can use *event channels* for mediating event messages between sources and sinks. An event channel is an entity responsible for registering subscriptions of a specific type of event, receiving events, filtering events according to specific modes, and routing them to the interested sinks.

A component can contain itself a finite number of other interacting components, called *sub-components*, allowing the components to be nested at an arbitrary level. In this case, it is said a *composite* component. At the lowest level, components are said *primitive*. Sometimes, assembling two components may require an adaptation of associated interfaces, whenever these interfaces cannot directly communicate for example. In this case, the adaptation is done with an extra entity, called *connector*, modelling the interaction protocol between the two components.

For each component model, a corresponding *Architecture Description Language (ADL)* allows to describe an assembly of components forming an application. From such a description, a set of tools are used to compile and generate the application code, while checking syntactical and even some semantical properties.

IV. Formalization

Our goal is to propose a new formalism for checking consistency of dynamic reconfigurations of component-based. For this purpose, we give first a set of concepts and then define our approach for checking consistency of CBS.

A. Concepts

1) Component-based configuration:

Definition 1. *A component-based configuration of a system S is defined as a triplet:*

$$Cg = \prec C, I, B \succ \quad where$$

- *C: is a set of components;*
- *I: is a set of interfaces;*
- *B: is a set of component connections or bindings.*

Definition 2. *A component c is defined as:*

$$c = \prec name, granul, state \succ$$

where:

- *name: is the unique name of the component C;*
- *granul: refers to granularity wich can be Composite or Primitive;*
- *state: is the current state of the component C, which can be Started or Stopped.*

Definition 3. *A component interface i is defined as:*

$$i = \prec itfc, role, visib, card, contig, sign \succ$$

where:

- *itfc: is the unique identifier of the interface (being of the form: component-name.interface-name);*
- *role: can be Client / Server (in the case of a service invocation interface) or Sink / Source (in the case of an event based interface);*
- *visib: refers to the visibility of the interface, which can be Internal or External;*
- *card: refers to the cardinality of the interface, which is Singleton or Collection;*
- *contig: characterizes the interface contingency, which may be Optional or Mandatory;*
- *sign: returns the interface signature.*

Definition 4. *A component binding b is defined with:*

$$b = \prec itfc - clt, itfc - srv \succ$$

where:

- *itfc-clt: refers to the invoking interface, and can be Client or Sink;*
- *itfc-srv: refers to the service interface, and may be Server or Source.*

2) Reconfiguration:

Definition 5. *Let be a configuration Cg_1 of a system S. We define a reconfiguration R of S, being in the configuration Cg_1, as an ordered set of primitive operations applied on Cg_1 :*

$$R = op_1, op_2, ..., op_n, n \geq 1$$

where $op_i, i = 1..n$, is one of the following reconfiguration operations:

1) *Delete a component*
2) *Add a component*
3) *Replace a component*
4) *Delete a binding*
5) *Add a binding*

The resulted configuration after application of R is denoted Cg_2.

We denote this by:

$$Cg_1 \overset{op}{\rightarrow} Cg_2$$

3) Predefined functions: To be able to specify constraints required for performing properly a reconfiguration, we need a set of predefined functions. For this objective, we propose the following functions:

1) CFather(cp) : returns the parent of the component cp;
2) CInterfaces(cp) : returns the interfaces list of the component cp;
3) CType(cp) : returns the type of the component cp;
4) IComponent(i) : returns the owner of the interface i;
5) IType(i) : returns the type of the interface i.

B. Constraints

To ensure the correction of a reconfiguration R applied on a system S, we define two sets of constraints:

- Constraints on primitive reconfiguration operations : Should be checked after each primitive operation.
- Global constraints : should be checked after the whole reconfiguration.

In the following, we specify these two sets of constraints.

1) Constraints on primitive reconfiguration operations: Let op be a primitive reconfiguration operation, applied on a configuration Cg_1 of a system S, resulting in the configuration Cg_2, where :

- $Cg_1 = \prec C_1, I_1, B_1 \succ$
- $Cg_2 = \prec C_2, I_2, B_2 \succ$

We denote this by:

$$Cg_1 \overset{op}{\rightarrow} Cg_2$$

In the following, we consider :

- A component : $cp = \prec name, granul, statut \succ$
- A binding : $b = \prec iclt, isrv \succ$

Primitive reconfiguration operations, applied on components cp and cp', are denoted as follows:

1) Delete a component cp :

$$del\_comp(cp)$$

2) Add a component cp :

$$add\_comp(cp)$$

3) Replace a component cp by anpther cp':

$$Repl\_comp(cp, cp')$$

4) Delete a binding b :

$$del\_bdg(b)$$

5) Add a binding b :

$$add\_bdg(b)$$

Table I gives the required constraints to be satisfied after each reconfiguration operation.

TABLE I: CONSTRAINTS ON PRIMITIVE RECONFIGURATION OPERATIONS

Operation	*Constraints*
del_comp(cp)	1) $C_2 = C_1 - cp$ 2) $I_2 = I_1 - CInterfaces(cp)$ 3) $\forall$ i $\in CInterfaces(cp), i \notin I_2$
add_comp(cp)	1) $C_2 = C_1 \cup cp,\ cp \notin C_1$ 2) $I_2 = I_1 \cup CInterfaces(cp)$ 3) $\forall i \in I_1, \exists j \in I_2\ tq:\ i.itfc = j.itfc$ 4) $\forall i \in CInterfaces(cp), i \notin I_1$
Repl_comp(cp,cp')	• $CType(cp)$ *is a subtype of* $CType(cp')$
del_bdg(b)	1) $B_2 = B_1 - b$ 2) $b \in B_1$
add_bdg(b)	1) $B_2 = B_1 \cup b$ 2) $b \notin B_1$ 3) $\exists\ b.iclt \in I_1 \wedge b.isrv \in I_1$ 4) $b.iclt.card$=$SINGLETON \Rightarrow$ $\forall$ b'$<\ iclt', isrv'\ >\in B_1, iclt' \neq iclt$ 5) $b.isrv.card$=$SINGLETON \Rightarrow$ $\forall$ b'$<\ iclt', isrv'\ >\in B_1, isrv' \neq isrv$

2) Global constraints: Let R be a reconfiguration that will be applied to a configuration Cg_1 of a system S, giving as a result a configuration Cg_2:

$$Cg_1 \overset{R}{\rightarrow} Cg_2$$

$$with:\ R = op_1, op_2, ..., op_n, n \geq 1$$

We specify the following constraints, which must be satisfied by Cg_2 :

1) $\forall$ b $\in B_2$, b.iclt.role = Client / Sink $\wedge$ b.isrv.role = Server / Source
2) $\forall$ b $\in B_2$, (b.iclt.contig = Mandatory) $\Rightarrow$ (b.isrv.contig = Mandatory)
3) $\forall$ b, b' $\in B_2$, b.iclt $\neq$ b'.iclt
4) $\forall$ b $\in B_2$, (CFather(IComponent(b.iclt)) = CFather(IComponent(b.isrv)))$\vee$ (b.iclt.visib = Internal $\wedge$ IComponent(b.iclt) = CFather(IComponent(b.isrv))) $\vee$ (b.isrv.visib = Internal $\wedge$ CFather(IComponent(b.iclt)) = IComponent(b.isrv))
5) $\forall$ i $\in I_2$ (i.role = Client $\wedge$ i.contig = Mandatory $\Rightarrow \exists!$ b $\in B_2$ tq: b.iclt = i)
6) $\forall$ b $\in B_2$, IType(b.isrv) $\subseteq$ IType(b.iclt)

C. Consistency of a configuration

Theorem 1. *A reconfiguration R, applied to a configuration Cg_1 of a system S, is* valid *if the resulting configuration Cg_2 satisfies all constraints defined on primitive reconfiguration operations and global constraints.*

Theorem 2. *A configuration Cg_i of a system S is* consistent *after a reconfiguration R if R is* valid.

V. ILLUSTRATION

To illustrate our approach, we use a navigator application similar to Mozilla already used in [20]. In such applications, components are usually equipped with an install manifest in XML format, allowing, among other things, to deliver the information needed to manage the version compatibility between components.

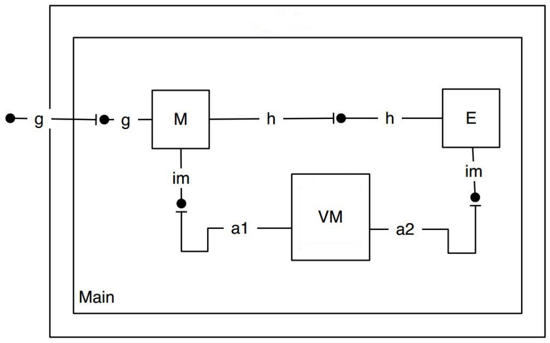


Fig. 1: Initial configuration

So, the architecture of the application consists of a composite component $MAIN$ composed of three primitive components (Figure 1):

1) M, the main application (e.g., Firefox);
2) E, an already installed plugin;
3) VM, a version manager component.

Each of the components M and E have an interface h with a signature H, being respectively a client and server interface. They also each have a server interface im of signature $InstallMf$. M has an additional server interface g of signature G, being the main interface exported to the global external interface of the application.

The Main composite exports business methods from M and supplies update, a control method implementing the upgrade operation. This method looks for a component with same id as E, having a more recent version and being compatible with M. In case of success, it replaces E with the new component.

Based on our formalization, we specify the initial configuration of Figure 1 as follows:

$$Cg_1 = \prec C_1, I_1, B_1 \succ$$

where:

- $C_1 = (Main, M, VM, E)$
- $I_1 = (Main.g, M.g, M.im, VM.a1, VM.a2, E.im, E.h)$
- $B_1 = (b1, b2, b3, b4)$

where:

- M = (M, Primitive, Started)
- VM = (VM, Primitive, Started)
- E = (E, Primitive, Started)
- Main.g = (Main.g, Server, External, Singleton, Optional, G)
- M.g = (M.g, Server, External, Singleton, Optional, G)
- M.h = (M.h, Client, External, Singleton, Optional, H)
- M.im = (M.im, Server, External, Singleton, Mandatory, InstallMF)
- VM.a1 = (VM.a1, Client, External, Singleton, optional, InstallMF)
- VM.a2 = (VM.a2, Client, External, Singleton, Optional, InstallMF)
- E.h = (E.h, Server, External, Singleton, Optional, H)
- E.im = (E.im, External, Singleton, Mandatory, InstallMF)
- b1 = (Main.g, M.g)
- b2 = (VM.a1, M.im)
- b3 = (VM.a2, E.im)
- b4 = (M.h, E.h)

When applying on this configuration a reconfiguration R, which removes the plugin E, we model this by the following reconfiguration R :

$$R = op_1, op_2, op_3$$

where:

- op_1 : $Del\_comp(E)$,
- op_2 : $Del\_bdg(b3)$,
- op_3 : $Del\_bdg(b4)$.

This resulted configuration is valid because it provides a new consistent configuration (given in Figure 2), which is defined as follows:

$$Cg_2 = \prec C_2, I_2, B_2 \succ$$

where:

- C_2 = (Main, M, VM)
- I_2 = (Main.g, M.g, M.im, M.h, VM.a_1, VM.a_2)
- B_2 = (b1)

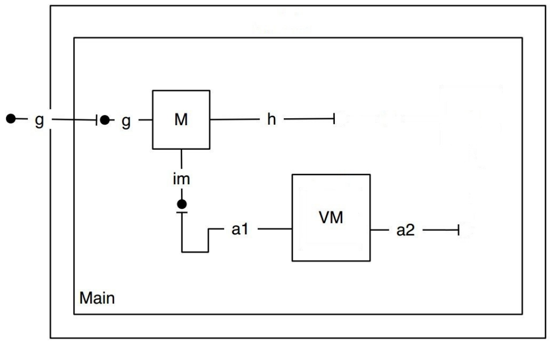


Fig. 2: Resulting configuration after reconfiguration

where:

- M = (M, Primitive, Started)
- VM = (VM, Primitif, Started)
- Main.g = (Main.g, Server, External, Singleton, Optional, G)
- M.g = (M.g, Server, External, Singleton, Optional, G)
- M.h = (M.h, Client, External, Singleton, Optional, H)
- M.im = (M.im, Server, External, Singleton, Mandatory, InstallMF)
- VM.a1 = (VM.a1, Client, External, Singleton, optional, InstallMF)
- VM.a2 = (VM.a2, Client, External, Singleton, Optional, InstallMF)
- b1 = (Main.g, M.g)
- b2 = (VM.a1, M.im)

By checking all defined constraints, we can say that R is valid. So, the new configuration $Cg2$ is consistent starting from the fact that $Cg1$ is consistent.

VI. CONCLUSION

In this paper, we presented a new formalism for checking consistency of dynamic reconfigurations of general component-based systems. For this purpose, we introduced formal concepts for modelling a component-based configuration and reconfiguration operations. We also defined required constraints that must be satisfied by the new configuration resulting after applying reconfiguration, to ensure consistency of the system.

Our approach can be instanciated to any existing component model, allowing thus genericity of the formalism.

Work is in progress to achieve automation of the proposed approach, by providing a toolbox based on the FOCALIZE programming environment [21]. This latter is based on a functional programming language with object-oriented features and allows to write formal specifications and proofs of designed programs. Proofs are build using the automated theorem prover Zenon [22] and Coq proof-assistant [23]. Future work also include modeling CBS before and after reconfiguration to allow quantitative analysis of CBS.

REFERENCES

[1] C. Szyperski, *Component software: Beyond Object-Oriented Programming*. Addison-Wesley Professional, 2002, vol. 2nd Edition.
[2] Sun Microsystems, "EJB 3.0 specification," http://www.oracle.com/technetwork/java/docs-135218.html, jul 2007.
[3] Object Management Group, "Corba component model (ccm) (CORBA) - specification, version 3.1, part 3: CORBA components," http://www.omg.org/spec/CORBA/3.3/Interoperability/PDF (November 2008), 2008.
[4] E. Bruneton, "Fractal tutorial," http://fractal.objectweb.org/tutorials/fractal/index.html (September 12 2003), September 2003.
[5] M. Leger, "Reliability of dynamic reconfigurations in component architectures," Ph.D. dissertation, Superior National School of Mines of Paris, 19 mai 2009.
[6] R. M. Vincenzo Grassi and A. Sabetta, "From design to analysis models: a kernel language for performance and reliability analysis of component-based systems," vol. 80, no. 11, 2005, pp. 25–36.
[7] O. M. Group, "(uml) unified modeling language (3rd release)," November 2004.
[8] Object Management Group, "UML unified modeling language (3rd release)."

[9] D. Freeman, *Markov chains*, Springer-Verlag, Ed. Springer-Verlag, 1983.

[10] L. Kleinrock, *Queueing systems. Volume I : Theory*, Wiley-Interscience, Ed. New-York: Wiley-Interscience, 1975.

[11] V. Grassi, R. Mirandola, and A. Sabetta, “A model-driven approach to performability analysis of dynamically reconfigurable component-based systems,” pp. 103–114, 2007.

[12] I. Warren, J. Sun, S. Krishnamohan, and T. Weerasinghe, “An automated formal approach to managing dynamic reconfiguration,” in *Automated Software Engineering, 2006. ASE ’06. 21st IEEE/ACM International Conference on*, september 2006, pp. 37–46.

[13] J. Hillman and I. Warren, “An open framework for dynamic reconfiguration,” in *Proceedings of the 26th International Conference on Software Engineering*, ser. ICSE ’04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 594–603.

[14] D. Jackson, “Alloy : a lightweight object modelling notation,” vol. 11, no. 2, Novembre 2002, pp. 256 – 290.

[15] L. T. David P.-C., “An aspect-oriented approach for developing self-adaptive fractal components,” 2006, pp. 82–97.

[16] E. Bruneton, T. Coupaye, M. Leclercq, V. Quéma, and J. Stefani, “The fractal component model and its support in java,” vol. 36, no. n. 11-12, 2006, pp. 1257–1284.

[17] M. Simonot and M. Aponte, “Formal modeling of control with fractal,” CEDRIC laboratory, CNAM-Paris, France, Tech. Rep. CEDRIC-08-1590, 2008, http://cedric.cnam.fr/index.php/publis/article/view?id=1590.

[18] V. Benayoun, “Fractal components with dynamic reconfiguration : formalization with focal,” 2008, http ://reve.futurs.inria.fr/.

[19] C. Szyperski, “Component technology - what, where, and how?” in *Proc. 25th Int. Conf. on Software Engineering*. IEEE, May 3–10 2003, pp. 684–693.

[20] M. Simonot and V. Aponte, “A declarative formal approach to dynamic reconfiguration,” pp. 1–10, 2009.

[21] INRIA and LIP6, “The focalize essential,” 2005, http://focalize.inria.fr/.

[22] D. D. R. Bonichon and D. Doligez, “Zenon : An extensible automated theorem prover producing checkable proofs,” vol. 4790, 2007, pp. 151–165.

[23] Y. Bertot and P. Casteran, *Interactive Theorem Proving and Program Development Coq Art: The Calculus of Inductive Constructions*. Addison-Wesley, 2004.

An Investigation on Quality Models and Quality Attributes for Embedded Systems

Lucas Bueno R. Oliveira
University of São Paulo - USP,
São Carlos, SP, Brazil
IRISA - Université de Bretagne-Sud,
Vannes, France
buenolro@icmc.usp.br

Milena Guessi
Dept. of Computer Systems,
University of São Paulo - USP,
São Carlos, SP, Brazil
guessi@icmc.usp.br

Daniel Feitosa and Christian Manteuffel
University of Groningen - RUG,
Groningen, The Netherlands
{d.feitosa, c.manteuffel}@rug.nl

Matthias Galster
University of Canterbury,
Christchurch, New Zealand
matthias.galster@canterbury.ac.nz

Flavio Oquendo
IRISA - Université de Bretagne-Sud,
Vannes, France
flavio.oquendo@univ-ubs.fr

Elisa Yumi Nakagawa
Dept. of Computer Systems,
University of São Paulo - USP,
São Carlos, SP, Brazil
elisa@icmc.usp.br

***Abstract*—Embedded systems have gained more and more importance in recent years, being adopted in a diversity of application areas. Due to the increasing variety and complexity of these systems, a rising demand for software quality can be observed. Initiatives proposing quality models and quality attributes (QM&QA) for embedded systems can already be found. Nevertheless, there is a lack of a complete, detailed panorama about the research that proposes QM&QA dedicated specifically to this domain. In this paper, we apply the systematic review technique to investigate how QM&QA for embedded systems have been defined, evaluated, and used. In addition, we identify which quality attributes are considered as the most important ones in the embedded systems domain. As a result, this work provides a detailed state-of-the-art about the QM&QA for embedded systems and identifies new, important research topics for the future, contributing to improve the quality of these systems.**

Keywords-Embedded System; Quality Model; Quality Attribute; Systematic Review.

I. Introduction

Recently, a large number of products containing embedded software has been developed and used, bringing an effective impact to the society. Embedded systems have been widely adopted in different application areas, such as telecommunication, transportation, entertainment, and medicine [1]. According to Liggesmeyer and Trapp [2], over the last 20 years, software's impact on the embedded system functionalities, as well as on the innovation and differentiation potential of new products, has rapidly grown. Besides that, the complexity and diversity of these products are creating a considerable challenge for embedded software development, which usually has to meet stringent requirements, such as real-time or performance [1]. The development process of embedded systems has to ensure the compliance with various quality attributes, such as maintainability, safety, security, and dependability. In this context, the quality assessment activity must be considered a key concern during the development of such systems. This statement is especially true considering the fact that many embedded systems are considered critical, i.e, systems whose failures may cause serious damage to the environment or to human lives, damage to expensive equipment, or non-recoverable financial losses [3].

In another perspective, software quality models have become well-accepted means to describe, manage, and predict software quality. Over the years, a variety of quality models have been proposed to support the development of general software systems. McCall's Quality Model [4], considered as the precursor of the actual models, establishes three major perspectives for defining and identifying the quality of a software product: product revision, product transition, and product operations. Each of these perspectives describes a set of quality attributes that refers to the ability of a software system to undergo changes, to adapt to new environments, and to adequately performs its functionalities. Similarly, Boehm's Quality Model [5] attempts to define software quality by a given set of attributes and metrics. Another important quality model is ISO/IEC 25010 standard [6], which incorporates quality goals that encompass a large number of quality attributes. Given its relevance, quality models and sets of quality attributes (QM&QA) that intent to specifically address the needs of embedded systems can also be found [7], [8]. These studies can be considered important initiatives, as embedded systems have particular characteristics, such as the use of dedicated hardware and real-time constraints, that differentiate them from general information systems. Nevertheless, as far as we are concerned, there is no complete, detailed view of how QM&QA have been defined, evaluated, and used in the embedded systems domain. Therefore, a study involving a broad, fair analysis of this research topic seems to be quite relevant, considering the impact that it could have on the quality of the embedded systems being developed.

The main objective of this paper is to present a detailed state of the art of QM&QA for embedded systems, the application areas that they are intended for, and how QM&QA have been evaluated. In addition, this work also aims at identifying which quality attributes are considered as the most relevant ones in the embedded systems context. For this, we have adopted and applied the systematic review technique [9], which allows for a complete, fair evaluation of a topic of interest. Results have shown that most studies are recent, indicating a growing interest and concern of the community on the proposition of QM&QA for embedded systems. Besides that, we have observed that there is a lack of quality models that are widely adopted and used by developers of embedded systems. Based on our findings, we intend that this state of the art makes it possible to identify interesting, important research topics for further investigations.

The remainder of this paper is organized as follows. Section II presents the conducted systematic review and describes its results. Section III presents the quality assessment of these results. Section IV summarizes the main, important findings of the systematic review and identifies perspectives of future research. Finally, Section V presents our conclusion and future work.

II. Systematic Review Application

Our systematic review was conducted from November/2012 to April/2013 by six persons: four software engineering researchers, an embedded system expert, and a systematic review specialist. To conduct our systematic review, we followed the process proposed by Kitchenham [9]. In short, this process is composed of three main phases: planning, conduction, and reporting. These phases are explained in more details during the presentation of our systematic review.

A. Phase 1 - Planning

In this phase, the objectives and the systematic review protocol are defined. The protocol consists of a predetermined plan that describes the research questions and how the systematic review will be conducted, i.e., the search strategy. It also establishes the selection criteria, the data extraction and synthesis method.

1) Research Questions: Aiming at finding possibly all primary studies to understand and summarize evidences about QM&QA for embedded systems, the following research questions (RQ) were established:

- **RQ1:** How are QM&QA for embedded systems defined?
  - **RQ1.1:** What are the information sources used to define QM&QA for embedded systems?
  - **RQ1.2:** Are the QM&QA developed in a prescriptive or descriptive manner?
- **RQ2:** What are the application areas where QM&QA for embedded systems have been used?
  - **RQ2.1:** Are the QM&QA designed for critical embedded systems?
  - **RQ2.2:** Which design approaches, such as service-orientation or component-orientation, have been adopted to develop these embedded systems?
- **RQ3:** How have QM&QA for embedded systems been evaluated?
  - **RQ3.1:** What is the level of evidence used to evaluate the QM&QA?
  - **RQ3.2:** In how many embedded systems the QM&QA have been applied?
  - **RQ3.3:** Have the QM&QA been used in actual projects?
- **RQ4:** What are the main quality attributes for embedded systems?

2) Search Strategy: In order to establish the search strategy and considering the research questions, we initially identified two main keywords “Embedded System” and “Quality Model”. We also identified related terms for these keywords: “Embedded Software”, “Quality Attribute”, “Non-functional Requirement”, “Non-functional property”, and “Quality Requirement”. We considered the plural form of all keywords and related terms. Besides that, only papers written in English were considered in our systematic review, since it is the most common language in scientific papers. We used the Boolean operator OR to link the main terms and their synonyms; furthermore, all these terms were combined using the Boolean operator AND. The final search string was: *(“Embedded System” OR “Embedded Systems” OR “Embedded Software”) AND (“Quality Model” OR “Quality Models” OR “Quality Attribute” OR “Quality Attributes” OR “Non-functional Requirement” OR “Non-functional Requirements” OR “Non-functional Property” OR “Non-functional Properties” OR “Quality Requirement” OR “Quality Requirements”).*

In addition to the search string, we also defined a control for our systematic review. For this, we considered two previously known studies [7], [8]. They were our baseline to check whether our search string was properly defined, i.e., if our string was able to find these studies in the publication databases. Moreover, in order to select the most adequate databases for our search, we considered the following criteria discussed in [10]: *content update* (publications are regularly updated); *availability* (full text of the primary study is available); *quality of results* (accuracy of the results obtained by the search); and *versatility export* (since much information is obtained through the search, a mechanism to export the results is required). The selected databases to our systematic review were: ACM [11], IEEE Xplore [12], ScienceDirect [13], Scopus [14], Springer [15], and Web of Science [16]. According to Dybå et al. [17] and Kitchenham et al. [18], these publication databases are the most relevant sources. Aiming at not missing any important primary study, we also considered the related works presented in the reference list of the primary studies selected by our systematic review.

3) Inclusion and Exclusion Criteria: The selection criteria are used to evaluate each primary study obtained from the publication databases. These criteria make it possible to include primary studies that are relevant to answer the research questions and exclude studies that do not answer them. Our inclusion criteria (IC) were:

- **IC1**: The primary study presents a quality model for embedded systems;

- **IC2**: The primary study reports the use of a quality model for embedded systems;
- **IC3**: The primary study proposes a set of quality attributes; and
- **IC4**: The primary study is an empirical study that has as outcome a set of quality attributes.

The established exclusion criteria (EC) were:

- **EC1**: The study does not propose or report QM&QA for embedded systems;
- **EC2**: The study is a previous version of a more complete paper about the same research; and
- **EC3**: The primary study is a table of contents, short course description, copyright form or conference proceedings.

4) Data Extraction and Synthesis Method: In order to extract data, we planned to build data extraction tables related to each research question. These tables will synthesize the results to facilitate drawing conclusions. During the extraction process, the data of each primary study will be independently extracted by two reviewers. In case of disagreements, discussions will be conducted. To summarize and describe the set of data, statistical synthesis method and meta-analysis will be applied.

B. Phase 2 - Conduction

In this phase, we adapted the generic search string defined in the Phase 1 according to the specificity of each publication database. The search of primary studies was then performed by searching for all primary studies that matched the adapted search string. After removing primary studies indexed by two or more publication databases, 308 primary studies remained for analysis. Initially, the title and abstract of each study were read and the selection criteria were applied. A total of 15 studies were selected for further reading. These studies were read in full by two reviewers and the selection criteria were again applied. As a result, nine primary studies were selected for the data extraction. Besides, we looked for the related work (i.e., the main references) of each primary study read in full. Among all related works evaluated, we selected two relevant primary studies that had not been previously identified [19], [20]. Finally, a set of 11 studies was selected as the most relevant to our systematic review.

Table I shows all primary studies included, their publication year, and references (Ref.). It is important to notice that only three primary studies found propose quality models for embedded systems (i.e., they were included by IC1). Therefore, most of studies are dedicated to provide sets of quality attributes for embedded systems. Moreover, it is possible to observe that 73% (i.e., 8/11) of the studies were published in the last five years, which might indicate an increasing interest for this topic of research.

C. Phase 3 - Reporting

This phase presents the analytical results of our systematic review. Data extraction and synthesis of knowledge considering each research question are discussed below.

TABLE I. QM&QA FOR EMBEDDED SYSTEMS

ID	Author	Year	Criteria	Ref.
S1	Wijnstra, J.G.	2001	IC3	[19]
S2	Purhonen, A.	2002	IC3	[21]
S3	Åkerholm, M. et al.	2004	IC4	[20]
S4	Choi, Y. et al.	2008	IC1	[22]
S5	Sherman, T.	2008	IC4	[8]
S6	Carvalho, F. and Meira, S.R.L.	2009	IC2	[23]
S7	Paulitsch, M. et al.	2009	IC3	[24]
S8	Peper, C. and Schneider, D.	2009	IC3	[25]
S9	Jeong, H.Y. and Kim, Y.H.	2011	IC1	[26]
S10	Guessi, M. et al.	2012	IC4	[7]
S11	Ahrens, D. et al.	2013	IC1	[27]

1) RQ1 - Research Question 1: This research question aims at understanding how QM&QA for embedded systems have been defined. For this, we have investigated which sources of information are most used to develop the QM&QA and whether they are defined in a descriptive or prescriptive way. Descriptive primary studies depict how quality has been addressed in systems of this domain. On the other hand, prescriptive primary studies introduce guidelines of how quality should be addressed in embedded systems. Table II summarizes the sources of information and methods of development used in each primary study.

We noticed that most of QM&QA for embedded systems (54.5%) were developed from documental analysis, i.e., using information collected in documents associated to existing systems, such as system requirement documents. Moreover, personal experience and literature reviews were considered in 36.4% and 27.3% of the primary studies, respectively. Developed systems, standards and regulations, interviews, questionnaires, existing software architectures, and on-going projects were also considered in at least one primary study. Furthermore, it is possible to observe that there is no predominance of prescriptive or descriptive studies. We also identified that there is no correlation between the information source and prescriptive/descriptive QM&QA. Thus, the choice of information sources may be more related to the context in which the model was defined than the purpose for what it was intended.

2) RQ2 - Research Question 2: This research question investigates for which application areas QM&QA for embedded systems have been developed. To answer this question, we collected data regarding the application areas of the embedded systems, as well as the approaches used to design these systems. We also collected data to discover whether QM&QA were designed to critical embedded systems. Table III summarizes the obtained results.

Regarding this research question, it is possible to point out that several studies (S5, S6, S9, and S10) are concerned about quality of embedded systems in general, i.e, without a specific application area. QM&QA for embedded systems for the transportation area can also be highlighted (S3, S7, and S11). With respect to the design approaches, we found out that they are often related to component-based embedded systems, as presented in studies S3, S4, S6, S9, and S11. It

TABLE II. INFORMATION SOURCES AND METHODS OF DEVELOPMENT USED TO DEFINE QM&QA

Source of information	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	(#)	(%)
Documental analysis		X		X	X	X			X		X	6	54.5
Personal experience	X							X	X		X	4	36.4
Literature review		X					X			X		3	27.3
Developed systems		X			X							2	18.2
Standards and regulations		X					X					2	18.2
Interviews		X									X	2	18.2
Questionnaires			X								X	2	18.2
Existing architectures							X					1	9.1
On-going project								X				1	9.1
Method of development	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	(#)	(%)
Prescriptive study		X	X	X		X		X				5	45.5
Descriptive study	X				X		X		X	X	X	6	55.5

TABLE III. APPLICATION AREAS AND DESIGN APPROACHES OF THE QM&QA

ID	Application Area	Design Approaches	Critical System
S1	Medical imaging	Product Lines	Yes
S2	Digital signal processing	Generic	No
S3	Automotive	Components	Yes
S4	Digital TV	Components	No
S5	Generic	Generic	Both
S6	Generic	Components	Both
S7	Aviation	Integrated Modular Architecture	Yes
S8	Ambient intelligence	Service Oriented Architecture	Yes
S9	Generic	Components	Yes
S10	Generic	Generic	Both
S11	Automotive	Components	Yes

TABLE IV. OVERVIEW OF THE EVALUATION OF QM&QA FOR EMBEDDED SYSTEMS

ID	Level of Evidence	Number of Systems	In use
S1	No evidence	NR	Yes
S2	Academic studies	0	NR
S3	Expert opinions or observations	NR	NR
S4	Industrial evidence	2	Yes
S5	No evidence	NR	NR
S6	Expert opinion or observations	0	NR
S7	Expert opinion or observations	NR	NR
S8	Demonstration or toy programs	1	NR
S9	No evidence	NR	NR
S10	Academic studies	NR	NR
S11	Industrial studies	1	NR

is also possible to identify QM&QA that are not limited to a specific type design approach, such as presented in studies S2, S5, and S10. Furthermore, we found out that most of primary studies (nine out of 11) are dedicated to critical embedded systems. This result was expected and reinforces the importance and interest in the quality of this type of systems.

3) RQ3 - Research Question 3: This research question investigates on the evaluation of the QM&QA for embedded systems available in the literature. For answering this question, we collected data about the level of evidence used in the evaluation, the number of systems in which these QM&QA have been applied, and whether they are in actual use or not. The following levels of evidence were considered: industrial evidence (i.e., actual use of QM&QA in industry), industrial studies (i.e., QM&QA developed in the industry); academic studies (e.g., controlled lab experiments or evidence based results); expert opinions or observations; demonstration or working out toy examples; and no evidence. Table IV presents the information about the evaluation of the QM&QA for embedded systems. Studies that do not report whether the proposal is in actual use or not are represented as Not Reported (NR).

It is possible to observe that only three studies (S4, S8, and S11) present QM&QA that were evaluated through their application in embedded systems. Five studies were evaluated using expert opinion (S3, S6, and S7) or academic studies (S2 and S10). Three studies do no present information about their evaluation (S1, S5, and S9). However, it is worth highlighting that QM&QA proposed in S1 and S9 are descriptive studies that emerged from personal experience (see Table II) and may not need an explicit evaluation. Besides that, it can be noticed that, among the QM&QA evaluated using embedded systems, only primary study S4 reports its application at least twice. Regarding the adoption of QM&QA, only S1 and S4 studies indicate that their proposals are currently supporting the evaluation of embedded systems. The other included studies do not present evidences about their current adoption. Despite these QM&QA may be in actual use, no publication reporting this information was found in our systematic review.

4) RQ4 - Research Question 4: This research question investigates the main quality attributes for embedded systems. Table V presents the main quality attributes identified in this review and the primary studies that address these attributes.

We identified 18 major quality attributes related to embedded systems. These attributes are those addressed by at least 25% of the primary studies, i.e., three or more studies. It is observed that the main quality attributes are related to maintainability and reliability. This result seems coherent, since an embedded system involves the coordinated project of software and hardware. Besides that, the maintainability is a challenging issue of the development of this type of

TABLE V. QUALITY ATTRIBUTES OF EMBEDDED SYSTEMS

Attributes	(#)	(%)	Primary studies
Maintainability	10	91	S1, S2, S3, S4, S5, S6, S7, S9, S10, S11
Reliability	10	91	S1, S3, S4, S5, S6, S7, S8, S9, S10, S11
Security	7	64	S3, S4, S5, S6, S7, S8, S9
Safety	7	64	S1, S3, S5, S6, S7, S10, S11
Functionality	7	64	S1, S4, S5, S6, S8, S9, S10, S11
Efficiency	7	64	S2, S3, S4, S5, S6, S10, S11
Portability	7	64	S1, S2, S4, S6, S9, S10, S11
Testability	7	64	S1, S3, S5, S6, S7, S9, S11
Performance	5	45	S2, S5, S7, S10, S11
Usability	5	45	S3, S4, S5, S6, S9
Availability	4	36	S1, S5, S9, S11
Extensibility	4	36	S1, S2, S3, S11
Reusability	4	36	S2, S4, S6, S11
Cost	4	36	S1, S2, S5, S6
Fault tolerance	3	27	S2, S9, S10
Recoverability/ Repairability	3	27	S6, S9, S11
Interoperability	3	27	S1, S9, S10
Flexibility	3	27	S3, S5, S6

systems. Embedded systems are also often used in safe-critical context and, therefore, they must be reliable. Most of studies also address security, safety, functionality, efficiency (i.e., efficient consumption of hardware resources, such as processor, memory, and battery), portability (i.e., ability of being transferred and used in a different environment), and testability as important quality attributes. Other quality attributes addressed by less the half of the studies were: performance, usability (i.e., ability of being understood, learned, configured, and used), availability, extensibility, reusability, fault tolerance, recoverability (repairability), interoperability, and flexibility.

III. Quality Assessment

In order to analyze the quality of the included primary studies, we developed a checklist containing seven questions based on the quality assessment created by Kitchenham et al. [28]. Table VI presents the quality assessment criteria and the scores obtained by the primary studies. For each question in the checklist, the following scale-point was applied: the study fully meets a given quality criterion (1 point), the study meets the quality criterion in some extent (0.5 point), and the study does not meet this quality criterion (0 point). Thus, the total quality score fell into the range between: 0 - 1.0 (very poor); 1.1 - 2.0 (poor); 2.1 - 3.0 (fair); 3.1 - 4.0 (average), 4.1 - 5.0 (good), 5.1 - 6.0 (very good), and 6.1 - 7.0 (excellent). It can be noticed that eight out of 11 studies were considered as having good quality. On the other hand, two studies were considered as having poor quality. Despite of that, these two studies were not excluded from this review because we were interested in covering all publications available in the research area. It is also important to highlight that studies considered as having poor quality did not present information about evaluation, limitation of their results, and perspectives of future research.

IV. Brief Discussion

After carrying out the systematic review, a first finding was that QM&QA are often defined using two or more different sources of information. This fact may evidence that the establishment of QM&QA is a complex task and requires broad knowledge about the domain. This review also points out that, among the studies that propose generic QM&QA (i.e., QM&QA that can be applied to any type of embedded system), only study S9 is described in the format of a quality model (i.e., included by IC1), but it is considered to have a poor quality. Therefore, contributions that provide widely accepted quality models for embedded systems are still necessary.

In parallel, QM&QA could be used as means to conduct quality evaluation of embedded systems. This review also pointed out that few QM&QA were evaluated using evidences obtained in the industry or in real embedded systems. Thus, more studies reporting experiences of evaluating embedded systems might increase the reliability of the QM&QA and also provide important feedback to improve them. In this scenario, this topic of research can be considered as a promising one and results of this review can be used as a starting point. Notice that the set of attributes can also be different, including a different distribution, if we considered specific application areas, such as automotive and robotics. Finally, we identified that only study S11 proposes a set of metrics related to its QM&QA. Therefore, we believe that the identification of metrics associated to QM&QA is also an important topic of research, and it can contribute to provide some measurement to the development of embedded systems.

V. Conclusion and Future Work

The adoption of quality models and the identification of most important quality attributes can contribute to improve the quality, which is so needed in embedded systems. In this perspective, the main contribution of this work is to present a detailed state of the art on the QM&QA available in literature, the way they were defined and evaluated, and the main quality attributes addressed by them. For this, we conducted the steps of a systematic review. As future work, we intend to make a more specific investigation of this research area, for instance, to identify metrics associated to each quality attribute. Furthermore, we intend to consolidate the results of this systematic review in a general quality model for embedded systems, aiming at contributing to a more effective development of such systems.

VI. Acknowledgments

This work is supported by Brazilian funding agencies São Paulo Research Foundation - FAPESP (Grant N.: 2011/06022-0 and 2011/23316-8), Capes/Nuffic (Grant N.: 034/12), and CNPq (Grant N.: 142099/2011-2, 474720/2011-0, and 239981/2012-0), as well as the INCT-SEC (Grant N.: 573963/2008-8 and 2008/57870-9).

TABLE VI. QUALITY ASSESSMENT OF THE INCLUDED PRIMARY STUDIES

Source of information	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11
Q1: There is a rationale for why the study was undertaken	1	1	1	1	1	1	1	1	1	1	1
Q2: It presents an overview about the state of the art of the area in which the study is developed	0.5	1	1	1	0.5	1	1	0.5	0.5	1	1
Q3: There is an adequate description of the context in which the work was carried out	0.5	0.5	1	1	1	0.5	1	1	0	1	1
Q4: It provides a clear justification about the methods used during the study.	0	0.5	0.5	0.5	0.5	0.5	1	1	0	1	0.5
Q5: There is a clear statement of contributions and has sufficient data been presented to support them	0	1	0.5	1	0.5	0.5	0.5	1	0.5	1	1
Q6: It discusses the credibility and limitations of their findings explicitly	0	0.5	0	1	0	0.5	1	1	0	1	1
Q7: It discusses perspectives of future works based on the study contributions	0	0.5	1	1	0	0.5	1	1	0	0	1
Study overall score	2	5	5	6.5	3.5	4.5	6.5	6.5	2	6	6.5
Study overall score (%)	29	71	71	93	50	64	93	93	29	86	93

REFERENCES

[1] W. Wolf, Computers as Components – Principle of Embedded Computing System Design, 2nd ed. Morgan Kaufman, 2008.

[2] P. Liggesmeyer and M. Trapp, "Trends in embedded software engineering," in IEEE Software, vol. 26, no. 3, 2009, pp. 19–25.

[3] A. Aguiar, S. Filho, F. Magalhaes, T. Casagrande, and F. Hessel, "Hellfire: A design framework for critical embedded systems' applications," in ISQED'10, San Jose, USA, 2010, pp. 730 –737.

[4] J. McCall, P. Richards, and G. Walters, Factors in Software Quality. Nat'l Tech. Information Service, 1977.

[5] B. W. Boehm, J. R. Brown, and M. Lipow, "Quantitative evaluation of software quality," in ICSE'76, San Francisco, USA, 1976, pp. 592–605.

[6] ISO/IEC, "Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models," ISO/IEC, Tech. Rep. 25010/2011, 2011.

[7] M. Guessi, E. Y. Nakagawa, F. Oquendo, and J. C. Maldonado, "Architectural description of embedded systems: a systematic review," in ISARCS'12, Bertinoro, Italy, 2012, pp. 31–40.

[8] T. Sherman, "Quality attributes for embedded systems," in Advances in Computer and Information Sciences and Engineering, T. Sobh, Ed. Springer, 2008, pp. 536–539.

[9] B. Kitchenham, T. Dybå, and M. Jørgensen, "Evidence-based software engineering," in ICSE'04, Edinburgh, Scotland, UK, 2004, pp. 273–281.

[10] O. Dieste, A. Grimán, and N. Juristo, "Developing search strategies for detecting relevant experiments," in Empirical Software Engineering, vol. 14, no. 5. Hingham, MA, USA: Kluwer Academic Publishers, 2009, pp. 513–539.

[11] ACM Digital Library, [Online]. Available: http://dl.acm.org/ - Accessed in 08/18/2013.

[12] IEEE Xplore, [Online]. Available: http://ieeexplore.ieee.org/ - Accessed in 08/18/2013.

[13] ScienceDirect, [Online]. Available: http://www.sciencedirect.com/ - Accessed in 08/18/2013.

[14] Scopus, [Online]. Available: http://www.scopus.com/ - Accessed in 08/18/2013.

[15] Springer, [Online]. Available: http://www.springerlink.com - Accessed in 08/18/2013.

[16] Web of Science, [Online]. Available: http://www.isiknowledge.com/ - Accessed in 08/18/2013.

[17] T. Dybå, T. Dingsoyr, and G. K. Hanssen, "Applying systematic reviews to diverse study types: An experience report," in ESEM'07, Madrid, Spain, 2007, pp. 225–234.

[18] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University and Durham University Joint Report, Tech. Rep. EBSE 2007-001, 2007.

[19] J. Wijnstra, "Quality attributes and aspects of a medical product family," in HICSS'01, Maui, Hawaii, 2001, pp. 1–10.

[20] M. Åkerholm, J. Fredriksson, K. Sandström, and I. Crnkovic, "Quality attribute support in a component technology for vehicular software," in SERPS'04, Linkoping, Sweden, 2004, pp. 1–9.

[21] A. Purhonen, "Quality attribute taxonomies for DSP software architecture design," in PFE'01, Bilbao, Spain, 2002, pp. 238–247.

[22] Y. Choi, S. Lee, H. Song, J. Park, and S. Kim, "Practical S/W component quality evaluation model," in ICACT'08, Phoenix Park, South Korea, 2008, pp. 259–264.

[23] F. Carvalho and S. Meira, "Towards an embedded software component quality verification framework," in ICECCS'09, Potsdam, Germany, 2009, pp. 248–257.

[24] M. Paulitsch, H. Ruess, and M. Sorea, "Non-functional avionics requirements," in Leveraging Applications of Formal Methods, Verification and Validation, 2009, vol. 17, pp. 369–384.

[25] C. Peper and D. Schneider, "On runtime service quality models in adaptive ad-hoc systems," in SINTER'09, Amsterdam, The Netherlands, 2009, pp. 11–18.

[26] H. Y. Jeong and Y. H. Kim, "A quality model of lightweight component for embedded system," in Applied Mechanics and Materials, vol. 121-126, 2011, pp. 4907–4911.

[27] D. Ahrens, A. Frey, A. Pfeiffer, and T. Bertram, "Objective evaluation of software architectures in driver assistance systems," in Computer Science - Research and Development, vol. 28. Springer-Verlag, 2013, pp. 23–43.

[28] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering - a systematic literature review," in Information and Software Technology, vol. 51, no. 1, Newton, MA, USA, 2009, pp. 7–15.

A Counter Rocket, Artillery and Mortar System with Laser Simulation Software

Maria Epp, Hendrik Rothe
Institute of Automation Technology,
Measurement and Information Technology
Helmut-Schmidt-University
Hamburg, Germany
maria.epp, rothe@hsu-hh.de

***Abstract* — Several countries develop laser weapons to use them for protecting critical infrastructures. Before the weapon is used for protection, the decision must be made, whether it is beneficial to use a laser instead of existing weapons. In order to make this decision, one must run several tests under varying conditions. These tests are not only very expensive, but also difficult to organize. This paper describes the Counter-RAM with laser simulation software. It simulates different attacks of rockets, artillery or mortar against the protected area. The simulation can simulate attack on the protected territory, the detection and tracking of missiles. It can classify the projectile as danger and simulate the intercepting of this projectile. This paper describes the development of the simulation.**

Keywords-C-RAM; simulation; RAM intercept; Laser weapon system

I. INTRODUCTION

Mortars and rockets are common weapons of insurgents. The inexpensive projectiles are fired at the defended area, where they can damage the infrastructure and kill or injure numerous people [1].

A Counter Rocket, Artillery and mortar system (C-RAM) is a defense system for providing warnings to vulnerable assets and for intercepting RAM threats in the air [2]. The system considered in this paper is equipped with a radar system and high energy laser weapons. Fig. 1 demonstrates an attack and engagement scenario: A launched mortar is detected by an acquisition radar. During the tracking, the trajectory of the projectile is predicted. If the projectile is classified as a target, a laser weapon is assigned to it. Once the laser is aligned to the target, optical tracking of the projectile is started. The laser weapon is activated and the target is destroyed in flight [3].

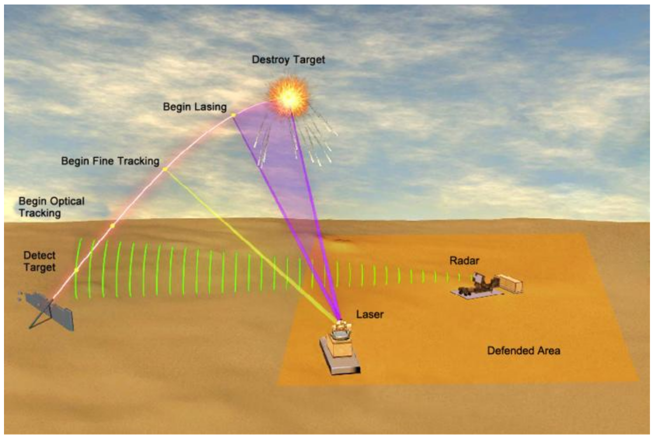


Figure 1: Typical engagement scenario (adapted from [3])

Only projectiles are modeled in the simulation, which are unguided and do not have sensors, that direct the flight path. The average flight time of such projectile is 25-35 seconds.

In the simulation, the projectile mass is concentrated at one point and is affected only by the force of gravity. Fig. 2 shows the simulated trajectory of a projectile: $\vec{v}_0$ is the muzzle velocity of the projectile, ϑ_0 its elevation, $\vec{F}_G$ is the force of gravity, $\vec{v}$ is the velocity, ω is the angle of sight.

The concept of the simulation is presented in [4]. Knapp and Rothe [4] describes the basic idea of a simulation for CRAM with laser weapon. This idea has been adapted and for this simulation and developed.

Section II describes the program flow of the simulation. The listing of the selected tools for programming can be found in Section III. Section IV will talk about possible advancements and extensions of the program.

II. SIMULATION DESCRIPTION

The simulation is split into two separate parts. The first part simulates an attack of a protected asset and the second part performs the defense of this area.

The software is divided into several individual modules.

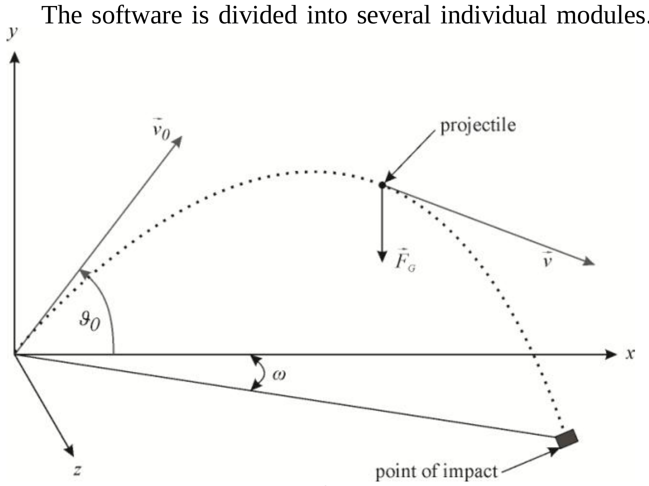


Figure 2: Trajectory of an unguided projectile

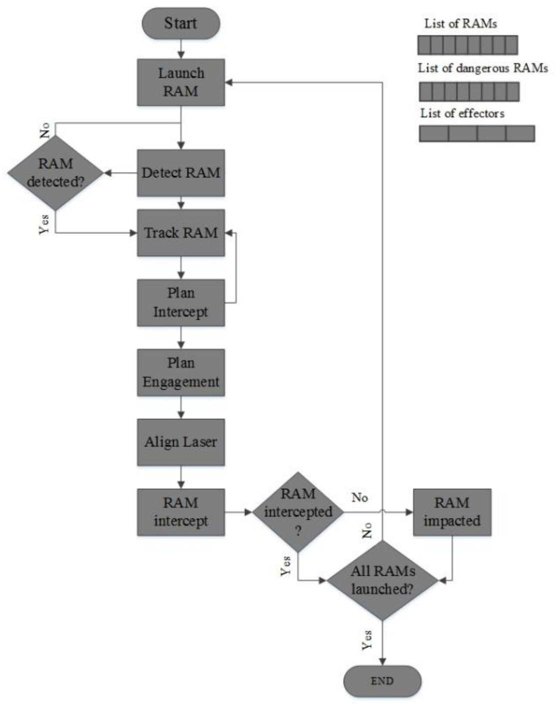


Figure 3: Workflow of the simulation

There is an internal database and a Graphic User Interface (GUI). A user has two options to start the simulation:

1) Enter all new relevant parameters for simulation via the GUI. Before the simulation starts, the parameters will be saved in the database.

2) Load existing simulation parameters. The parameters can be changed and saved as new simulation data or the old data set can be overwritten.

The workflow of the simulation is illustrated in Fig. 3. The simulation starts by firing of first projectile.

A. RAM Launch

This module simulates an attack on the protected area.

All RAMs are saved in a list and are sorted by firing time. At the beginning of the simulation, the trajectory for each projectile in the list is calculated. As of now, only vacuum trajectories are taken into account in the simulation.

The (x, y, z) –coordinates of the projectile, with respect to time, are calculated using equations (1), (2), and (3) [5].

$$x = \vec{v}_0 t \cos \vartheta_0 \quad (1)$$

$$y = \vec{v}_0 t \sin \vartheta_0 - {}^1\!/_2\, g t^2 \quad (2)$$

$$z = x \sin \omega \quad (3)$$

Once the first RAM in the list is launched, the simulation is started.

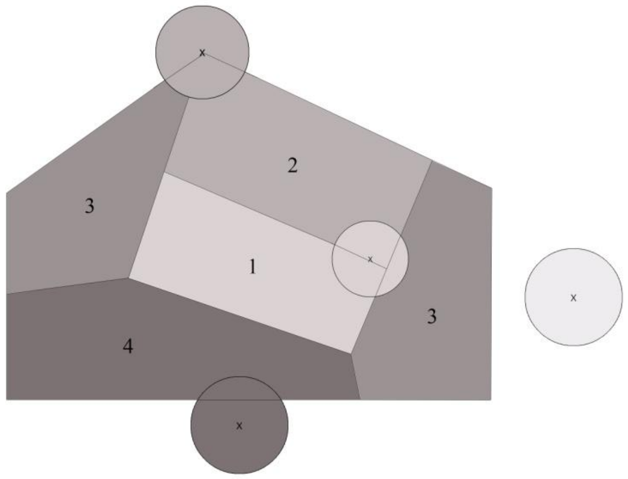


Figure 4: Defended area and impact points of the projectile with bursting radius: 1-4 are priorities of districts. 1is highest, 4 is lowest priority.

B. RAM Detection

Hence, the simulation of the defense of the asset begins.

The detection radar is located in the middle of the area, which is to be defended. The radar has the angle of sight [°], the radar range [m], and the detection rate [s]. An arbitrary number of RAMs can be detected by the radar. To determine whether the projectile is inside the radar range, the distance between the radar position (x_r, y_r, z_r) and the current position of the projectile (x_p, y_p, z_p) is calculated by using equation (4). The calculated distance is compared to the radar range. If the distance is less than the radar range, the projectile is classified as detected.

$$d = \sqrt{(x_r + x_p)^2 + (y_r + y_p)^2 + (z_r + z_p)^2} \quad (4)$$

C. RAM Tracking

As soon as a projectile is detected, the tracking of its trajectory starts. The tracking rate is defined by the user. An arbitrary number of projectiles can be tracked simultaneously.

The radar data is used in the trajectory prediction model [6, 7]. During tracking, the time and coordinates of the impact point are calculated [4], recurrently.

The prediction model needs several tracking datasets to calculate the trajectory. Therefore, only the projectile is tracked at first. Once enough data is available the first prediction coordinates and prediction times are determined. The more tracking data is available, the more accurate the predicted impact point will be calculated.

D. Interception Planning

Fig. 4 shows a defended area, which is split into several districts. Each district has a different priority. As soon as the first predicted impact point has been calculated, it is decided whether the projectile is a threat for the protected area. It is determined, whether the calculated point of impact, with its bursting radius taken into account, is inside the defended zone. If the bursting radius is located outside of the area, it is discarded. If the radius intersects with districts of several priorities, the highest priority will be assigned to the threat level of the projectile.

If more than one projectile is classified as a threat, these will be saved in a list and sorted by threat level.

E. Engagement Planning

If more than one projectile is identified as a threat, it must be decided in which order the projectiles should be fought. In the simulation software, the user has the choice between three engagement principles:

1) *Interception by first in first out (FIFO) principle:* The first detected projectile will be intercepted first.

2) *Interception by intercept time:* The projectile with the shortest intercept time will be intercepted first.

3) *Interception by priority of projectile:* The projectile with the highest priority in the list will be intercepted first.

F. Laser Assignment

After the order of engagement is determined, a laser weapon must be allocated to each projectile.

The weapon assignment problem is a fundamental problem of battle management. The problem is to assign weapons to RAM-threats in an optimal way. The expected damage to the protected area has to be minimized [8, 9].

At that moment, the weapon, that can be directed to the target the fastest, will be chosen. During the engagement analysis, the laser weapon is assigned to a threat. The laser can be assigned to another threat as long as it is not activated. Once the laser is activated, it is blocked for other threats. The projectile is destroyed within 3-8 seconds of engagement time, depending on distance to the threat. Because the more is the projectile to the laser weapon, the longer laser takes to heat the threat, due to the scattering of laser light.

G. Program End

The program terminates, once all projectiles from the list of RAMs have been processed. The number and the fighting time of the intercepted projectiles are displayed and recorded. For the impacted projectiles in the protected area, the fraction of damage is calculated. For each of the laser weapons, the consumed energy is recorded.

III. Simulation Design

1) Time continuous simulation: The program is based on time continuous simulation to make the simulation realistic. Thereby, the variables of the program are varied at defined points of time. At the beginning, the user can define the time step of the simulation via the GUI module. Each module is started at defined time steps.

2) Material:

a) C#: This is an object-oriented programming language, which has been developed by Microsoft. It is a widespread programming language which is why there are many internet communities to help with the programming, and also there is good support from Microsoft. Visual Studio 2012 [10] has been chosen as the programming environment.

b) NUnit: This is an open source unit testing framework from Microsoft. NUnit tests the modules of computer programs for correct functionality.

c) Windows Presentation Foundation (WPF): The GUI is developed with WPF [11]. It is a graphic framework for Windows-based applications. WPF uses DirectX [11]. WPF is based on the Extensible Application Markup Language (XAML).

IV. Conclusion

Each module is tested with NUnit individually. Once the modules are fully functional, the next step is to combine all modules and to test common systems with several different scenarios.

The existing simulators combat the approaching targets with artillery [12, 13]. The David's Sling System is the state of the art by C-RAM systems, based on interception by artillery [14]. The in this paper described system simulates the fighting of RAMs with laser weapon.

After a number of tests have been performed, the statement must be made whether or not it would be beneficial to adopt the laser weapon as a defense weapon.

References

[1] M. Libeau, "Laser Counter Rocket, Artillery, and Mortar (C-RAM) Efforts," NAVAL SURFACE WARFARE CENTER DAHLGREN DIV VA, January 2012, pp. 82–85.

[2] C. Corbett, B. Beigh, and S. S. Thompson, "Counter-rocket, artillery, and mortar (C-RAM) joint intercept capability: shaping the future joint force," Fires, March-April 2012, pp. 46–54.

[3] J. Schwartz, G. T. Wilson, and J. Avidor, "Tactical High Energy Laser," SPIE Proceedings on Laser and Beam Control Technologies, vol. 4632, January 21, 2002.

[4] M. Knapp and H. Rothe, "Concept for Simulating Engagement Strategies for C-RAM Systems using Laser Weapons," Conference on Defense and Military Modeling & Simulation. San Diego, CA, 2012.

[5] D. E. Carlucci and S. S. Jacobson, "Ballistics: Theory ans Design of guns and ammunition," CRC Press, 2008, pp. 195-202.

[6] M. Graswald, I. Shaydurov, and H. Rothe, "Analysis of weapon systems protecting military camps against mortar fire," Computational ballistics III, Southampton: WIT Press, 2007, pp. 21-30.

[7] I.Shaydurov and H. Rothe, "Hitting with the first shot: miniaturized fire control computer with digital signal processors" Duesseldorf: Wiedemeier & Martin, vol. 17, 2007, pp. 32-36.

[8] R. K. Ahuja, A. Kumar, K. J. James, and J. B. Orlin, "Exact and Heuristic Methods for the Weapon Target Assignment Problem," MIT Sloan School of Management, Working Paper 4464-03, July 2003.

[9] A. Toet and H. de Waard, "The Wepon-Target Assignment Problem," TNO Human Factors Research Insitut, February, 1994.

[10] Microsoft, http://www.microsoft.com/visualstudio/eng/products/visual-studio-ultimate-2012, 2013

[11] Microsoft, http://msdn.microsoft.com/library/vstudio/ms754130, 2013

[12] M. Knapp, M. Graswald, and H. Rothe “Simulation for Detrmining Engagement Strategies for C-RAM systems,” Proceeding of the 2011 Summer Computer Simulation Conference, 2011, pp. 168-174

[13] M. Graswald, I. Shaydurov, and H. Rothe. "Analysis of weapon systems protecting military camps against mortar fire," Computational Ballistics III, Ashurst (2007).

[14] “David’s Sling System – First Successful Interception Test,” Israel Defense, http://www.israeldefense.com/?CategoryID=483&ArticleID=1784 , 2012.

Towards Cloud-based Collaborative Software Development: A Developer-Centric Concept for Managing Privacy, Security, and Trust

Roy Oberhauser
Computer Science Dept.
Aalen University
Aalen, Germany
roy.oberhauser@htw-aalen.de

Abstract—**Cloud-centric collaboration in (global) software development is gaining traction, resulting in new development paradigms such as Tools-as-a-Service (TaaS). Yet both within and between clouds, there are associated security and privacy issues to both individuals and organizations that can potentially hamper collaboration. In this paper, an inter-cloud security and privacy concept for heterogeneous cloud developer collaboration environments is described that pragmatically addresses the distributed collection, storage, transmission, and access of events and data while giving individuals fine-granularity control over the privacy of their collected data. In a case study, the concept was implemented and evaluated by adapting an existing collaborative development and measurement infrastructure, the Context-aware Software Engineering Environment Event-driven framework (CoSEEEK). The results showed its practicality and technical feasibility while presenting performance tradeoffs for different cloud configurations. The concept enables infrastructural support for privacy, trust, and transparency within teams, and can support compliance with privacy regulations in such dynamic collaborative environments.**

Keywords-cloud-based software engineering environments; cloud-based software development collaboration; global software development; privacy; security; trust

I. INTRODUCTION

Global software development (GSD) [1] is increasingly taking advantage of cloud-based software applications and services [2] and realizing its collaboration potential. Data acquired and utilized during the software development and maintenance lifecycle is no longer necessarily locally controlled or even contained within an organization, but may be spread globally among various cloud providers with the acquired data retained indefinitely. Tools-as-a-Service (TaaS) [3] and cloud mashups will enable powerful new applications that utilize the acquired SE data [4]. And while the technical landscape is changing, the corporate landscape is also. A 2005 survey of American corporations conducted by the American Management Association showed that 76% monitored employee Internet connections, 50% stored and reviewed employee computer files, and 55% retained and reviewed email messages, with a rapidly increasing trend [5].

The ability to measure and minutely observe and track software developers during their work is becoming technically and economically viable to employers, managers, colleagues, virtual teams, and other entities. While metrics can be useful for personal improvement (cp. Personal Software Process), abuse is also possible (consider misuse of public profiling). While software services and apps for the public typically attend to user privacy due to their longevity, mass accessibility, and legal scrutiny, relatively little attention has been paid to the privacy needs of software developers, an estimated 17 million worldwide [6].

Consequently, privacy is becoming a looming concern for software developers that faces unique technical challenges that affect collaboration: it involves a highly dynamic technical environment typically at the forefront of software technology and paradigms (e.g., new languages, compilers, or platforms), uses diverse tools ([3] identifies 384) and heterogeneous project-specific tool chains (e.g., application lifecycle management, version control systems, build tools, integrated development environments, etc.), it is project-centric (unique, short-lived undertakings), it may involve multinational coordination (offshoring), etc.

Yet the trust climate plays a vital role in the success of virtual and distributed teams [7], and trust and transparency are considered vital values for effective teams and collaboration [8][9]. Where trust exists (cp. Theory Y [10]), collected data can be utilized collaboratively to enhance team performance [10], for instance by utilizing event data to coordinate and trigger actions and to provide insights, whereas where data is misused as an instrument of power, monitoring, or controlling (cp. Theory X [10]), individuals require mechanisms for protection. Since the technical development infrastructure cannot know a priori what trust situation exists between some spectrum of complete trust to complete distrust, infrastructural mechanisms should support collaboration within some spectrum, while allowing the individuals and organizations to adapt their level of data transparency to the changing trust situation.

Privacy is control over the extent, timing, and circumstances of sharing oneself. Cloud service users currently have few personal infrastructural mechanisms for retaining and controlling their own personal data. Diverse privacy regulations are applicable within various geographic realms of authority. Various (overlapping) (multi-)national laws and regulations may apply to such (global) collaborative cloud contexts. For instance, Germany has a Federal Data Protection Act, the European Union has a Data Protection Directive 95/46/EC, and within the United States, various states each have their own internet privacy laws.

Many privacy and security principles are typically involved including notice, consent, disclosure, security, earmarking, data avoidance, data economy, etc. Various challenges for security and privacy in cloud environments remain [11][13]. In the interim, pragmatic infrastructural approaches are needed to deal with the issues in some way.

The main contribution of this paper is to elucidate requirements for and describe a solution concept that pragmatically addresses various privacy and security concerns in cloud-based dynamic heterogeneous collaborative development environments (CDE). It is based on service layering, introduces distributed cloud-based datasteading for individuals, and mediates trust with brokers. Its technical feasibility and performance tradeoffs were investigated in a case study.

This paper is organized as follows: the next section details and provides some justification for the assumptions and requirements for a solution, and Section 3 describes related work. In Section 4, the solution concept is introduced, and the following section provides details of a technical implementation based on the concept. Evaluation results are presented in Section 6, which is then followed by a conclusion and description of future work.

II. REQUIREMENTS

The following requirements, assumptions, or constraints (denoted by the prefix R: in italics) were elicited from the primary problems, goals, and challenges introduced in the preceding section, and considered to be generally applicable for any conceptual solution. They are summarized here to highlight key considerations in the solution concept.

Multi-cloud configurability (*R:MCC*): private cloud (*R:PrC*), public cloud (*R:PuC*), and community cloud (*R:CoC*) support for a wide array of deployment options. *Provider-specific cloud API independence* (*R:PAI*) to support wide applicability and avoid provider lock-in. *Cloud compatibility* (*R:CCO*) with current public cloud provider and private cloud APIs and services, meaning no exotic solutions requiring special configurations that would limit usage. *Single tenancy* (*R:ST*) in the personal (developer's) cloud to reduce risk (e.g., to avoid a misconfiguration compromising a much larger set of tenants simultaneously) and avoid access by an organizational administrator, which involves an additional trust issue.

Disclosure (*R:D*): three fundamental levels of shall be supported: non-disclosure, anonymized disclosure, and personally-identifiable disclosure to specific requestors. *Sensor Privacy* (*R:SP*): It is assumed that any client-side and server-side sensors, (e. g., version control system sensors) distribute personally-identifiable events according to a privacy concept. *Entity-level privacy control* (*R:EPC*): granularity of privacy is controllable by any and all entities involved (persons, organizations).

Restricted network access (*R:RNA*) to collaboration participants, e.g., via Virtual Private Networks (VPN), to reduce cloud accessibility to collaborators only. *Secure communication* (*R:SC*) to protect internal data transmission, even within a VPN for personal privacy. *Basic security mechanisms* (*R:BSM*): Reliance on widely-available off-the-shelf security mechanisms (e.g., HTTPS), without any dependence on specialized or exotic hardware or software security platforms (e. g., Trusted Platform Module) or research-stage mechanisms that would constrain its practicality. Beyond (*R:SC*), e*ncryption* (*R:ENC*) can protect data accessibility and storage.

Trusted code implementation (*R:TCI*): Open source and/or independent code audits together with secure distribution mechanisms (e.g., via digital signatures from a trusted website) provide assurance that the code implementation can be trusted. Additionally, remote runtime *code integrity verification* (*R:CIV*) shall be supported to allow agents (e.g., automated temporally random auditing requests or manually initiated user requests) to detect any tampering with the implementation, sensors, configuration, or the compromise of any privacy safeguards.

In summary, a primary tenet is that organizations and teams want to support privacy freedom for individuals, support and value self-organizing teams, and not hinder electronic collaboration and communication. While together these requirements are in no way sufficient or complete, they nevertheless provide a practical basis and can be useful for furthering discussion.

III. RELATED WORK

In the area of global software development, [3] discusses support for TaaS and [14] Software-as-a-service in collaborative situations. Neither go into detail on various privacy issues, nor is support for various aforementioned requirements, e.g., individual (*R:EPC*). Examples industrial offers for cloud-based collaboration include Atlassian OnDemand and CollabNet CloudForge. Individual (*R:EPC*, *R:D*) do not appear to be supported.

Work on more general multicloud collaboration includes [4], which similarly supports opportunistic collaboration without relying on cloud standardization based on the use of proxies. However, aspects such as (*R:BSM*, *R:CI, R:EPC*) were not considered and a technical implementation was not investigated.

Work in the area of standardization and reference architecture includes [15], which mentions privacy but fails to prescribe a solution. [16] lists various security and interoperability standards and their status, but their maturity and market penetration considering (*R:MCC*) and (*R:CCO*) remain issues.

Various general cloud security mechanisms have been proposed. Privacy as a Service (PasS) [17] relies on secure cryptographic coprocessors to provide a trusted and isolated execution and data storage environment in the computing cloud. However, its dependency on hardware within cloud provider infrastructure hampers (*R:PAI*). Data protection as a service (DPaaS) [18] is intended to be a suite of security primitives that enforce data security and privacy and are offered by a cloud platform. Yet this would inhibit (*R:PAI*). Other work such as [19] describe privacy-preserving fine-grained access control and key distribution mechanisms, but are not readily available for a pragmatic approach that is usable today (*R:BSM*).

IV. SOLUTION CONCEPT

For a cloud-based context-aware collaboration system to have satisfactory utility, it will depend on some type of event and data collection and communication facilities. Thus this foundational infrastructure should be equipped with basic trust and security mechanisms, such that upper level services such as context-awareness and collaboration can ensue.

Thus, to provide a flexible solution for such environments, a primary principle in the solution is the application of the *Service Layer* design pattern to provide a decoupling and separation of concerns shown in Figure 1. The lower conceptual Event and Data Services Layer includes event and/or data services for an entity (person/team/organization) including acquisition, storage, retention, and dissemination, while the upper Collaboration and Tools Services Layer includes CDE and tool services that utilize lower layer data to provide collaboration, data sharing, analytics, and other value-added services over which an entity may have more limited privacy control mechanisms.

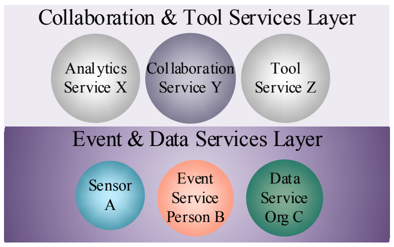


Figure 1: Services Layer Pattern

A second solution principle is the introduction of a *datastead*, shown in Figure 2, which is loosely analogous to the concept of homesteading or seasteading. In this case, an individual (or some unit) manages and controls clearly delineated data resources in the cloud for which they have or receive responsibility and ownership rights. The technical implementation of a datastead can be in the form of a personal cloud in the case of an individual, or within a private cloud for an organization. The third principle is the inclusion of a *Trust Broker* that mediates between service and data access, acting as both a cloud service broker (for interoperability with various tools) and cloud security broker (for security). Akin to the Trusted Proxy pattern [20] and Policy Enforcement Point [20], it constrains access to protected resources and allows custom, finely-tuned policies to be enforced (*R:EPC*). Rules can be used to configure and distinguish/filter access by event types, timeframes, projects, etc. It provides secure communication mechanisms (*R:SC*) to authenticate and authorize data acquisition and data dissemination in the datastead, as well as interoperability mechanisms for various collaboration and tool services. Only client requests from preconfigured known addresses are accepted. A management interface to the Trust Broker provides the datastead owner with policy management capabilities. It also supports data anonymization on a per request basis if so configured. For secure storage, the Trust Broker encrypts (*R:ENC*) acquired events and data (Encrypted Storage pattern [20]) to prevent unauthorized access by administrators or intruders, and protects access to the encrypted storage typically on a single port (Single Access Point pattern [20]). The Trusting Broker supports runtime code integrity (*R:CI*) via remote attestation, and a client, called the Trusting Tool, can be invoked periodically or event-based to ensure that the Trust Broker has not been tampered with.

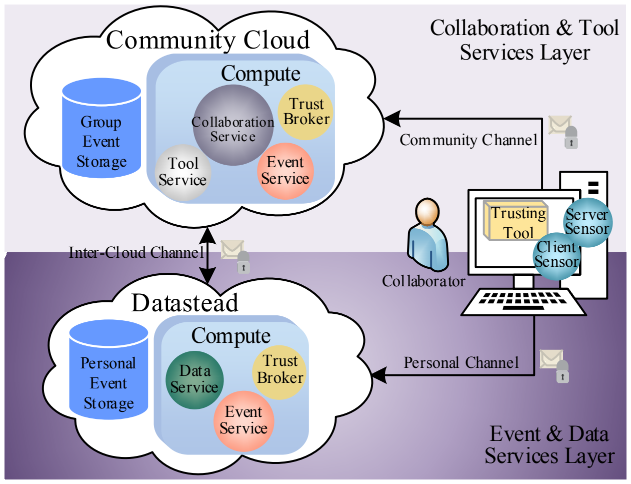


Figure 2: Generic Solution Concept

Secure Channels and Secure Sessions [20] are used to protect the transmission between the sensors and the datastead (the Personal Channel), between sensors and the Community Cloud (Community Channel), as well as between the datastead and any collaboration and tool services (Inter-cloud Channel). For a community cloud, a VPN is used to limit network access.

V. TECHNICAL IMPLEMENTATION

To determine the technical feasibility of the solution concept and provide a concrete case study, the solution concept was applied to an existing heterogeneous CDE called the Context-aware Software Engineering Environment Event-driven framework (CoSEEEK) [22], which had hitherto not incorporated privacy or security techniques. CoSEEEK's architecture and integrated technologies are shown in Figure 3. Its suitability is based on its portability (use of mainly Java and web-based languages), non-commercial access to the code and dependent technologies, its reliance on distributed communication mechanisms, and its heterogeneous tool support.

For event acquisition, CoSEEEK relies on the Hackystat framework [22] and its SE tool-based sensors (e.g., Ant, Eclipse, Visual Studio) for event extraction and event storage (shown in red in Figure 1). Hackystat does not currently provide extensive security and privacy mechanisms. For an insight, [24] briefly describes some of its security issues.

Service Layer Separation: the Hackystat-related elements (shown in red) were hereby separated into the Event and Data Services Layer and the remaining elements were placed in the Collaboration and Tools Services Layer.

Cloud configuration: To meet (*R:MCC*, *R:CCO*, *R:PAI*), two different cloud platforms were utilized in isolation. To represent a public IaaS cloud provider configuration (*R:PuC*), Amazon Web Services (AWS) was used, using Elastic Compute Cloud (EC2) for computing services, the Elastic Block Store (EBS) for storing configuration files and XML database, and the Relational Database Service (RDS) which holds the sensorbase.

To represent a private cloud (*R:PrC*) or community cloud (*R:CoC*) deployment, OpenStack was used with Compute used for computing and Object Storage used in place of EBS storage; and since nothing directly equivalent to AWS RDS was available, a Compute instance with Object Storage that contains a MySQL Server database was configured. Single tenancy (*R:ST*) with one Compute instance per developer with access restricted to the developer was configured.

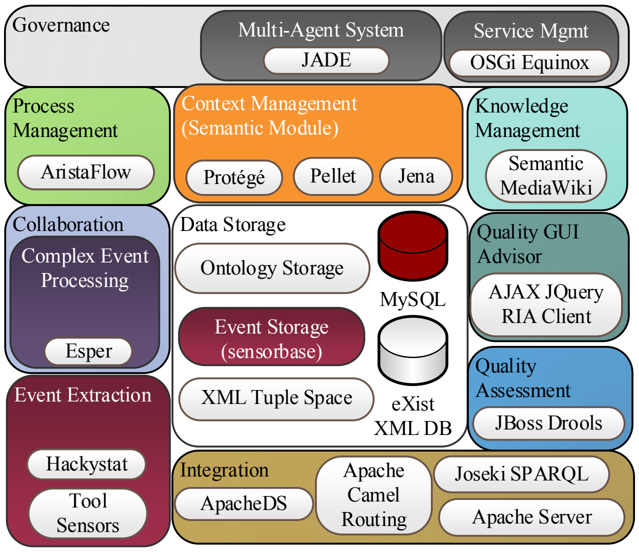


Figure 3. CoSEEEK Architecture (affected areas shown in red).

Trust Broker: the Trust Broker supports (*R:D*) was implemented in Java. The open source Restlet Framework for Java SE was used to provide the REST-based interface. An example of a query that can be sent is the following, specifying the project via the sensorbase_id, the timeframe, the sensor data type, the tool, and its uri source.

```
   GET
/trustbroker/sensordata/{sensorbase_id}?
startTime={startTime}&endTime={endTime}&
sdt_name={sdt_name}&tool={tool}
   &uriPatterns={uriPatterns}
```

Encryption of events (*R:ENC*) can be optionally configured. For encryption of arriving events and decryption of events on authenticated and authorized retrieval, Java's AES 128 and the SHA-256 hash algorithm were used (*R:BSM*). One reason for encrypting the storage is that it provides an additional form of protection, should, e.g., a provider's agent or intruder gain access.

The measurement database called sensorbase in Hackystat required a few minor adaptations. For (*R:D*), to support anonymization the HACKYUSER table was extended to include an anonymization flag, which is checked before responding, replacing a userid with anonymous. In order to support HTTPS connections, the sensorbase client (*R:SP*) was modified and rebuilt, requiring any sensors to utilize this modified jar file. HTTPS (*R:BSM*) was used to secure all three communication channels (personal, community, and inter-cloud) (*R:SC*). Additional properties were added to indicate the location of the keystore. SSH was used to configure and manage each cloud. Security groups were used in both AWS and OpenStack.

To implement remote attestation, on the client-side, a user configures the Trusting Tool with the expected checksum value (provided e.g., by the admin or a trusted website), version, and the interval for rechecking. On the service side, a REST interface sensorbase/checksum was added that loads the local adapted sensorbase.jar file, computes the SHA-256 hash value using `java.security.MessageDigest`, and returns this value and the sensorbase version to the Trusting Tool. While not foolproof, since any unauthorized access on the server or client could allow spoofing, it provides an additional level of confidence. Various stronger jar file tampering technologies could be employed if needed, such as componio JarCryp bytecode encryption.

VI. EVALUATION

The case study evaluated the technical feasibility of the concept based on the technical implementation. However, security and privacy are highly contextually dependent on the expectations, requirements, environment, risks, policies, training, available attack mechanisms, implementation details (bugs), configuration settings, etc., making a comprehensive formal assessment in this area difficult. So the assumption is made that the prescribed privacy and security mechanisms suffice or are balanced for current developer needs in developer settings.

Since CoSEEEK is a reactive system, the ability to respond adequately to contextual changes via events is highly dependent on network latency; the evaluation focuses on this area for various cloud settings.

As to hardware, the Client PC (for use by a developer) has an i5-2410M (2.3-2.9 GHz) dual core CPU and 6GB RAM with 32-bit Windows XP SP3. The network consists of gigabit Ethernet and two 1 Gbit connections from the university campus in Germany to the Internet Provider.

Representative for a private (*R:PrC*) or community cloud where a datastead could also be placed, the OpenStack configuration (OSCfg) consisted of a local intranet server with an i5-650 (3.2-3.4GHz) dual core CPU, 8GB RAM, and 64-bit Ubuntu Server 12.04. The OpenStack Cloud Essex Release was installed on the Server via DevStack and the Compute instances also ran Ubuntu Server 12.04. MySQL v. 5.5.24 was used for Hackystat sensorbase storage in a Compute instance.

As a public cloud provider (*R:PuC*) representative, a free AWS configuration (AWSCfg) was chosen. It consisted of t1.micro EC2 instance types located in US-EAST-1d (Virginia) with 613 MiB memory, up to 2 EC2 units (for short periodic bursts) with low I/O performance running 64-bit Ubuntu Server 12.04. MySQL v. 5.5.27 was used for the Hackystat sensorbase storage in AWS RDS.

Common software included Hackystat 8.4 with the Noelios Restlet Engine 1.1.5 and JDK 1.6.

Typical network usage scenarios were considered, thus no optimizations were applied to any configurations nor was an artificially quiet network state created. All results are the average of 10 repeated measurements (with one exception noted below). A secure configuration denotes using the TrustBroker via HTTPS (*R:SC*) with encrypted storage (*R:ENC*), and an insecure configuration means no TrustBroker and using HTTP. VPN (*R:RNA*) overheads were not measured.

To determine delays from the client to the datastead in cloud variants, on the client PC Ant was invoked, causing the Hackystat Ant sensor to send one XML event to the Server (a write in the remote sensorbase) consisting of 235 bytes of event data and 73 bytes of overhead. As shown in Figure 4, the average network latency using an insecure OSCfg was 214 ms, for a secure OSCfg 389 ms, and for a secure AWSCfg 608 ms.

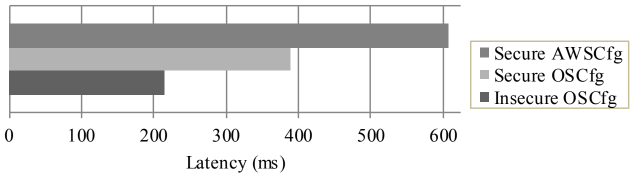


Figure 4: Latency (in ms) for sending an event (33 bytes) from the client PC to the server sensorbase.

Once events are in the datastead, then latencies between computing instances in a cloud are of interest, since the collaboration or tool services will be retrieving this data (shown in Figure 5 and Figure 6). For AWSCfg, a single query for 67 events (15818 bytes) between two EC2 instances took 78 ms on average via HTTP and 84 ms over HTTPS. In a secure configuration the retrieval took 347 ms. For OSCfg between two Compute instances, a single query took 38 ms to return 22 events (5243 bytes). Note that HTTP insecure reads in the private cloud had two anomaly values (178 and 210 ms) that would have changed the average from 38 to 69, and were also far larger than any secure value measurements. Thus, these 2 measurement values were removed, and the average created from the remaining 8 values. These large latencies could perhaps be attributed to a network, disk, operating system, or OpenStack related issue. Continuing with the measurements with 39 events (9238 bytes), HTTPS requests took 60 ms while in the secure configuration it averaged 61ms.The overhead of the privacy approach is the addition of SSL, brokering a second SSL connection, and encryption. For the OSCfg, the difference of TrustBroker and decryption showed on average only a 1ms difference to that with purely SSL. One explanation could be that the extra overhead is minimal compared to the data transfer delays between OpenStack instances, but further investigation of OpenStack internals and profiling would be required.

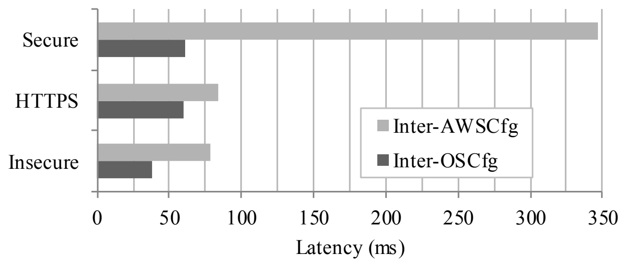


Figure 5: Private vs. public cloud inter-computing instance query latencies grouped by security (in ms)

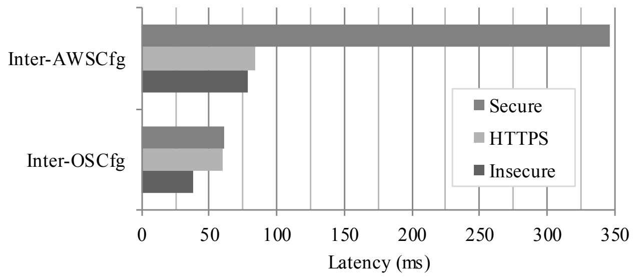


Figure 6: Inter-cloud query latency grouped by cloud type for different degrees of security (in ms).

Based on the results shown in the above figures, the use of the secure configuration of the OSCfg within a private or even a community cloud setting would appear to have acceptable performance overhead for cloud-centric collaborative development work, and distributed retrieval from datasteads is viable for responding to changes in the collaborative situation. On the other hand, the use of the secure configuration in the public cloud (AWSCfg), as shown in this perhaps worst case as a no cost offshore minimal public cloud setting, incurs substantially higher network latencies. Obviously choosing geographically close locations when possible is recommended. Also, provisioning sufficient computing and I/O resources support to deal with the additional inter-cloud and security mechanism overheads would also reduce such lags in public cloud configurations. Optimization in this area would also be promising.

To determine the remote attestation overhead, the Trusting Tool was measured on the PC using the AWSCfg over SSL. The average request-response latency was 702 ms. On the server, this involved loading and calculating the SHA-256 hash value for the 5.5 MB large sensorbase.jar file. Thus the attestation mechanism of the remote cloud instance could be configured to be automatically invoked periodically by client-side sensors at regular intervals in a separate thread or process to not interfere with other network communication.

In summary, the evaluation showed that network latencies incurred by the concept are most likely insignificant for collaboration in PrC settings, but that security overheads in global PuC settings may require optimization attention.

VII. Conclusion and Future Work

To address security and privacy in collaborative cloud development, this paper presented a practical concept with entity-level control of non-, anonymized-, and personally-identifiable disclosure for multiple cloud configurations. It can further both collaboration and trust by giving individuals transparency and control and allowing them to adjust disclosure to the changing trust situation. The paper contributes a practical basis for illustrating issues, eliciting awareness, community discussion, and and may increase self-regulation and infrastructural privacy offerings. Organizations adopting such a privacy infrastructure show that they value and trust their employees, enabling them to reap mutual trust rewards. Also, one could envision, for instance, that an audited "we don't spy here" seal might help attract and retain developers.

The evaluation showed its technical feasibility and practicality, requiring only minimal adaptation of the CoSEEEK CDE. The Trust Broker enables fine granularity access control to personal data. Performance was sufficient in private cloud configurations, while public cloud configurations using additional security and privacy mechanisms may require optimization to ensure fluid collaboration situational response.

Limitations and risks include: extending privacy/trust support within and across collaboration layer tools, non-detection/discovery of (un)intentionally unspecified/hidden sensors, data manipulation risk by datastead owners themselves, and provider-side access or manipulation risk. For service provider trust issues, building your own datastead cloud server site could be considered.

Future work can consider the inclusion of various data provenance and data integrity mechanisms to mitigate manipulation risk. In the face of shifting privacy norms, infrastructural support for data confidentiality is needed to limit disclosure of distribution data beyond its original intent, like lifetime constraints, transitivity bounds, and claims-based access [25]. Enhanced remote attestation mechanisms could be investigated. Since service privacy is also a broader issue, development and adoption of global industry service privacy standards combined with independent privacy audits involving all service layers would enhance trust of cloud-based data acquisition and usage offerings.

Acknowledgment

The author acknowledges Jürgen Drotleff for his conceptual, implementation, and measurement contributions.

References

[1] S. I. Hashmi et al. (2011, August). Using the Cloud to Facilitate Global Software Development Challenges. In Global Software Engineering Workshop (ICGSEW), 2011 Sixth IEEE International Conference on (pp. 70-77). IEEE.

[2] R. L. Grossman, "The case for cloud computing," IT professional, 11(2), pp. 23-27.

[3] M. A. Chauhan and M. A. Babar, "Cloud infrastructure for providing tools as a service: quality attributes and potential solutions," In Proceedings of the WICSA/ECSA 2012 Companion Volume, ACM, 2012, pp. 5-13.

[4] M. Singhal et al., "Collaboration in Multicloud Computing Environments: Framework and Security Issues, " Computer, 46(2), IEEE Computer Society, New York, 2013, pp. 76-84.

[5] G. D. Nord, T. F. McCubbins, and J. H. Nord, "E-monitoring in the workplace: privacy, legislation, and surveillance software," Communications of the ACM, 49(8), 2006, pp. 72-77.

[6] M. Parsons, "The challenge of multicore: a brief history of a brick wall," EPCC News, Issue 65, University of Edinburgh, 2009, p. 4.

[7] T. Brahm and F. Kunze, "The role of trust climate in virtual teams," Journal of Managerial Psychology, 27(6), 2012, pp. 595-614.

[8] A. C. Costa, R. A. Roe, and T. Taillieu, "Trust within teams: The relation with performance effectiveness," European journal of work and organizational psychology, 10(3), 2001, pp. 225-244.

[9] T. DeMarco and T. R. Lister, Peopleware. Dorset House, 1987.

[10] D. McGregor, The Human Side of Enterprise. McGrawHill, New York, 1960.

[11] B. Al-Ani and D. Redmiles, "Trust in distributed teams: Support through continuous coordination," IEEE Software, IEEE Computer Society, 26(6), 2009, pp. 35-40.

[12] P. Louridas, "Up in the air: Moving your applications to the cloud," IEEE Software, 27(4), IEEE Computer Society, New York, 2010, pp. 6-11.

[13] H. Takabi, J. B. Joshi, and G. J. Ahn, "Security and privacy challenges in cloud computing environments," IEEE Security & Privacy, IEEE Computer Society, 8(6), 2010, 24-31.

[14] R. Martignoni, "Global sourcing of software development-a review of tools and services," In Fourth IEEE International Conference on Global Software Engineering (ICGSE 2009), IEEE Computer Society, 2009, pp. 303-308.

[15] F. Liu et al., NIST cloud computing reference architecture. NIST Special Publication, 500, 292, 2011.

[16] M. Hogan, F. Liu, A. Sokol, and J. Tong, Nist cloud computing standards roadmap. NIST Special Publication, 35, 2011.

[17] W. Itani, A. Kayssi, and A. Chehab, "Privacy as a service: Privacy-aware data storage and processing in cloud computing architectures," In Eighth IEEE International Conf. on Dependable, Autonomic and Secure Computing (DASC'09), IEEE Computer Society, 2009, pp. 711-716.

[18] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud data protection for the masses, " Computer, 45(1), 2012, pp. 39-45.

[19] M. Nabeel and E. Bertino, "Privacy-Preserving Fine-Grained Access Control in Public Clouds," Data Engineering, 21, 2012.

[20] D. M. Kienzle, M. C. Elder, D. Tyree, and J. Edwards-Hewitt, Security patterns repository version 1.0. DARPA, 2002.

[21] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, Security Patterns: Integrating security and systems engineering. Wiley, 2006.

[22] G. Grambow, R. Oberhauser, and M. Reichert, "Enabling Automatic Process-aware Collaboration Support in Software Engineering Projects," In Software and Data Technologies, Springer, Berlin Heidelberg, 2013, pp. 73-88.

[23] P. M. Johnson, "Requirement and Design Trade-offs in Hackystat: An In-Process Software Engineering Measurement and Analysis System," Proc. 1st Intl. Symposium on Empirical Software Engineering and Measurement, 2007, pp. 81-90.

[24] P. M. Johnson, C. A. Moore, J. Miglani, and S. Zhen, Hackystat design notes. 2001.

[25] D. Reed, D. Gannon, and J. Larus, "Imagining the future: Thoughts on computing," Computer, 45(1), 2012, pp. 25-30.

Two-Dimensional Models' Processing Using Principles of Knowledge-Based Architecture

Andrejs Bajovs, Oksana Nikiforova
Faculty of Computer Science and Information Technology
Riga Technical University
Riga, Latvia
e-mail: andrejs.bajovs@rtu.lv, oksana.nikiforova@rtu.lv

***Abstract* — Presently, the technological diversity increases the attention to Model Driven Software Development, which provides system modeling at the high level of abstraction and further generation of software components. In this aspect, the task of the automatic code generation starts to play an important role and requires a new generation of the research directed to the quality of model and model transformation result. This paper discusses an ability to use several principles of artificial intelligence and knowledge management and offers so called knowledge-based architecture for code generation from the Unified Modeling Language class diagram and a verification of a class diagram itself.**

Keywords- UML class diagram; code generation; knowledge base; model verification

I. INTRODUCTION

An increasing impact of the role of system modeling during software development facilitates the leading positions of Object Management Group (OMG) [1] and its solution for system abstraction, modeling, development, and reuse – Model Driven Architecture (MDA) [2]. A key component of usages of MDA is Unified Modeling Language (UML) [3], which defines several kinds of diagrams, their elements and notation. UML diagrams describe the system from different aspects: static diagram represents system structure, dynamic diagrams represents system behavior. Fully automated transformation of system model, defined at platform independent level into platform-specific source code, is the main goal of MDA.

Currently, this goal has not yet been achieved completely, due to problems with definition of system dynamic aspects and their translation into code components [2]. But even description of system static elements would give a good initial preparation for system development and its further refinement with dynamic aspects. This static system representation in the form of UML class diagram and further generation of software components could replace significant amount of routine work performed by programmers during software development. Reducing its amount could give developers an opportunity to focus on more important tasks, thus helping to improve the quality of computer systems' developing process.

Model-Driven Architecture defines that the system's models could be automatically transformed from one level of abstraction into another. These levels involve not only graphical, but also textual models, including a source code. So, according to MDA, a graphical model could be automatically transformed into a source code. Such transformation process is commonly called code generation.

The idea of automatic code generation is not new. The first code generators were compilers which appeared in the middle seventies and used text-to-text generation techniques [4]. Since then, a significant amount of different standards appeared to support the idea of automatic code generation, however the practical side of this field was left almost untouched. Nowadays, a significant amount of different tools exists, which implement the most popular code generation approach – text templates. However, the authors' previous study shows that the code generation as a result of the UML class diagram transformation is of a low quality [5]. As designed for the concrete situations (thus, required to be frequently rewritten), templates, possibly, limit the functionality of some popular code generators. The other problem is that the code generators do not "think" like a human while doing their job and should be endowed with means of at least artificial intelligence.

Therefore, authors state code generation as an object to research and propose knowledge-based code generator architecture, which allows not only generating the source code, but also verifies the correctness of a model and thus a model transformation result.

The goal of the paper is to describe how the basic principles of artificial intelligence could be used to increase the quality of the code generation process. This paper specifies the background of the term "code generation" and reveals the related problems. In order to solve them, the hypothesis of the knowledge-based code generator architecture is described. In addition, the small practical example is presented to reveal the essence of the proposed theory.

The paper is structured as follows. The second section describes the roots of code generation and related problems which disturb its evolution. Section three introduces the knowledge-based code generator architecture and describes its parts, advantages, and disadvantages. The mechanism of how the introduced architecture works is explained in section four. The fifth section gives an overview of the researches related to the code generators, which use artificial intelligence. Section six concludes the paper.

II. Code Generation: State of the Art

The term code generation has several interpretations. One of them is defined by OMG's MDA. It states that implementation of a concrete target platform is generated from a model containing the target platform's specific details using pre-defined and tool supported transformations. Actually, OMG did not invent anything new, but standardized older framework – Model Driven Software Development (MDSD) [6]. Both of MDA and MDSD are related to a term "model", which according to [1] is *"... a description or specification of a system and its environment for some certain purpose.*" However, MDA considers models to be central in the development process (assuming that the model represents a set of diagrams that express the whole software system) [7]. According to MDA, these diagrams are used to build the systems for any platform, however MDSD does not claim such portability at all. In contrast of MDSD, MDA suggests using only UML diagrams to describe the system at a high level of abstraction. In general, MDA is more strict than MDSD, which allows much more ways of building the computer system by using models [6].

There are four basic models for systems' development proposed by MDA: computation independent, platform independent, platform specific and implementation specific model. The first one reflects to business and its models. The next two represent analysis and detailed design models of software system to be developed. The last one reflects to implementation and runtime models and, in fact, it is a system's source code. MDA also defines that each of the described models could be transformed into the others [8]. This paper focuses on the automatic transformation of platform specific model to the implementation specific model.

While the OMG organization was developing theoretical basis of the research area, practical side of code generation started to fall behind. Nowadays, a significant amount of different standards related to code generation exists [9], but no methods could completely describe how to apply all these theory into practice. The problem is that OMG invented their standards for templates and transformation languages, but almost forgot about looking at the core process of code generation itself.

Speaking about theory, the computer science describes two different code generation approaches [10], but both of them involve word mapping to model elements. In addition, the study from [5] shows that some of the nowadays most popular code generators are not producing a good quality code because of lack of smart ways to verify correctness of the models.

Authors are making experiments with different software development environments and different tools, positioned as MDA/MDD support tools [5], and have detected several inadequacies between expected code and code generated by the tool. Unfortunately, the current experiments with modeling tools that generate program code from UML class diagram show a weak and unsatisfactory results compared to the expected. Authors have identified a number of problems, which can be generally divided into two groups:

- Modeling tools allow to create improper element constructions and use incompatible keyword connections that leads model transformation into incorrect code, that can`t be compiled.
- Generated code does not correspond to notation and details used in model, which leads to loss of information in the result code.

The root problem is in the simplicity of program code generators, which just transfer the pattern of model information into the program code without any additional testing and decision making on the required information conversion for the target programming language. Generators do not have any additional knowledge support about target platform restrictions, laws and keyword combination. Some tools like SPARX Enterprise Architect [11] have code template editor with built-in transformation templates, which can be modified to support custom needs, but this does not solve the problem of the lack of base information about target platform, because restrictions might be needed for combination of elements and not one-to-one element mapping. The second mentioned group points to the complexity of the generators negligence. The result program code does not represent appropriate constructions for semantics used in the model, resulting in loss of information and devalue of the work invested to provide additional details in the model.

It means that it is not enough with simple word mapping, and machine should be taught to apply some knowledge performing code generation. Inspired by this idea, in the next sections authors propose their hypothesis of applying some principles of artificial intelligence in code generation process to supplement it with the model verification.

III. The Knowledge-Based Code Generator Architecture

In this section, authors propose the hypothesis of the knowledge-based code generator architecture, which is shown in Fig. 1.

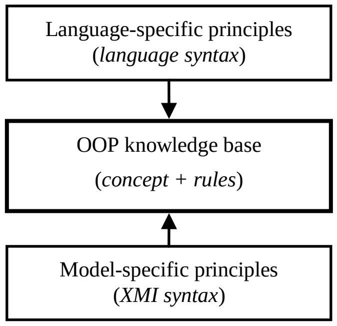


Figure 1. Knowledge-based code generator architecture.

The reason authors call it "knowledge-based" is as follows. As it was mentioned before, code generation is nothing but model transformation to code performed by computer. But how do human beings act, while transforming models to source code? It could differ from concrete individual, but commonly, each element of the model is taken and transformed, in a step-by-step manner, into code according to some knowledge of the model's notation, programming language syntax and fundamental rules of object-oriented paradigm. In authors' opinion, the word "*knowledge*" is the keyword here. That is the reason why the proposed architecture consists of three blocks: Object-Oriented Programming (OOP) knowledge base, model-specific principles and language-specific principles. All of them are explained in the next subsections of the paper.

A. Description of the knowledge-based architecture's blocks

The main block of the proposed architecture is OOP knowledge base, which describes the field of object oriented programming in a high level of abstraction. It represents only the very basics and does not describe anything connected with the concrete programming languages, models or platform-specific things. This is expressed in a way of ontology [12], which keeps two main things: conceptual information about OOP and basic rules to support validating the correctness of the UML class model.

The first is represented as a tree structure, which shows the relationships between different concepts of OOP (e.g., class, visibility, attribute, method, etc.). The simple example of such structure is shown in Fig. 2. Due to the complexity of the OOP itself, the relations between some concepts (visibility and attribute/method, type and name, interface and method, etc.) are omitted at the example to make it more readable and simpler to understand.

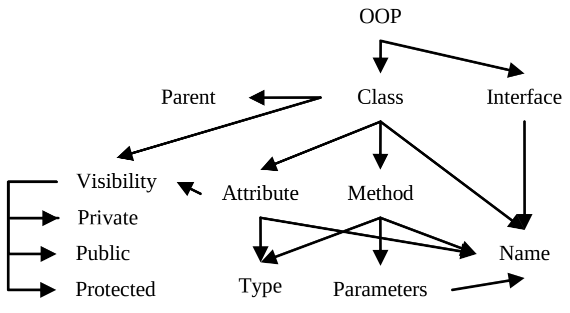


Figure 2. Example of OOP structure.

The second part of the OOP knowledge base is an alternative to Object Constraint Language (OCL). This block is represented by the set of rules, where each rule is a first-order logic (predicate) expression. This set of rules describes some restrictions which exist in the context of OOP (e.g., attribute can be only one at a time: private or public or protected).

The research of [5] defines some of the rules which are most commonly missed by code generators. They are:

1. If class contains at least one abstract method, then it must be marked as abstract;
2. A non-abstract class that is derived from an abstract class must include implementations of all inherited abstract methods;
3. Because an abstract method must be overridden in the derived class, then it must not be private;
4. While overriding an abstract method, the access modifier ought to be the same as for the overridden base method, e.g., if it is public, then in the derived class it can not be protected, because it must be public.

The rules mentioned above could be formally expressed in the way shown in Fig. 3.

1. has(Class, Method) & abstract(Method) → abstract(Class);
2. ¬abstract(Class1) & parent(Class1, Class2) & abstract(Class2) & inherited(Method, Class1, Class2) → overriden(Method, Class1, Class2);
3. abstract(Method) & overriden(Method) → ¬private(Method);
4. overriden(Method1, Method2) & abstract(Method2) → equals(visibility(Method1), visibility(Method2).

Figure 3. Formal expression of the model validation rules

The other block of the knowledge-based architecture is a set of language-specific principles or in other words, the syntax of different programming languages. In fact, there are several sets of such rules – each represents concrete programming language. The description of the syntax should be similar to Backus-Naur Form (BNF) notation [13] because its level of formalization allows to be easily interpreted by computer. The syntax of languages should be described using templates which associate concepts from the OOP knowledge base with its formal syntax. Although templates have some major disadvantages [5] which force to find alternatives to replace them, it is preferable to use them here. However, in this context templates should be maximally laconic and structured, describing the whole syntax of a concrete programming language rather than a particular case. The example of a simplified description of a Java class is shown in Fig. 4.

```
"class" <Name> [<Parent>]
["implements"  <Interface>
{","<Interface>}] "{"
          [{<Attribute>}]
"}"
```

Figure 4. Example of the class syntax description using BNF notation

Such markups as <*Name*> or <*Parent*> are taken from the OOP concept (see Fig. 2). During code generation the <*Name*> is replaced by the name of a particular class while <*Attribute*> is replaced by another piece of code which in case of Java is defined like this:

[<*Visibility*>] [<Scope>] <*Type*> <*Name*> [= <*Value*>]*;*

As it was stated earlier, the BNF notation is used to specify the syntax. Thus, blocks which are enclosed inside "*{}*" are repeating blocks, but blocks inside "*[]*" are those which can not be in the code for it to be correct, etc. A word inside "<>" points to a concrete block of the syntax which is associated with the concrete OOP concept. The last is a modification which is used for proposed architecture and is not connected with BNF.

The third block includes the model-specific principles that, in fact, represent the mapping of the concepts from the OOP knowledge base to the Extensible Markup Language Metadata Interchange (XMI) representation of the model [14]. This should be done using slightly extended XPath language [15]. Since various modeling tools implement XMI format differently [16] this block contains various sets of described mappings which are specific to the XMI format of the concrete tool. For example, let us assume the concepts shown in Fig. 2, which are mapped to the Extensible Markup Language (XML) document shown at Fig. 5.

```
<model>
   <class id = “1” name = “A” p_id = “2”>
     <attributes>
       <attribute  visibility  =  “private”
type = “int”>
           <name>A_atr1</name>
       </attribute>
     </attributes>
   </class>
   <class id = “2” name = “B”></class>
</model>
```

Figure 5. Example of the class syntax description using BNF notation

In this case, a mapping of the Class concept could be done as *//class*, which means that this concept takes its data from the *class* XML element. The name of the class in turn could be accessed as *<element>/@name* . *<element>* is a reserved directive which points to the element being iterated before a current one since all concepts make a hierarchical structure. This means that the knowledge-based code generator takes a full path *//class/@name* to access the name of the class.

It is also important that language-specific and model-specific principles' blocks could include overloading of some of the classic OOP rules from the OOP knowledge base according to the concrete programming language or model. Section IV shows how all of these blocks work together.

B. *Analysis of the knowledge-based architecture*

The proposed architecture does not have an ability to autonomously derive code as logical consequence of the knowledge-base like advanced AI code generators do. Basically, the approach does the standard template-based model-to-code transformation where additional intelligence is reflected into using such fundamental AI structures as ontology and first-order logic rules. Thus, ontology, syntax description and rules proposed by the authors could be represented as the equivalent of MDA meta-model, OCL and the templates, but their specter of appliance is wider, as well as they are more universal. For example, OCL is designed directly for UML and is much more oriented on constraining values rather than the structure of the models. In contrast, predicate rules do not depend on any concrete syntax so they could constrain every model by working directly with the essence of OOP itself. As for the proposed templates, they have less complex structure and focus on describing language's syntax rather than simple XMI mapping.

The main advantage of the proposed code generator architecture is its precise structure. Knowledge-based architecture defines the exact set of tasks for each of its blocks. It also specifies different levels of abstraction for describing contents for its blocks. The architecture gives an opportunity to split block creation tasks between different independent specialists where each of them should work on concrete task at a specific level of abstraction. Moreover, the OOP principles are a kind of bridge between a model and a programming language. This means that theoretically, each of the templates can be used with each of the model-specific principles. Rewriting or adding new ones also do not affect the opposite part. In addition, OOP knowledge base is the bridge which stands between the problem and solution domain. This is reflected in Fig. 6.

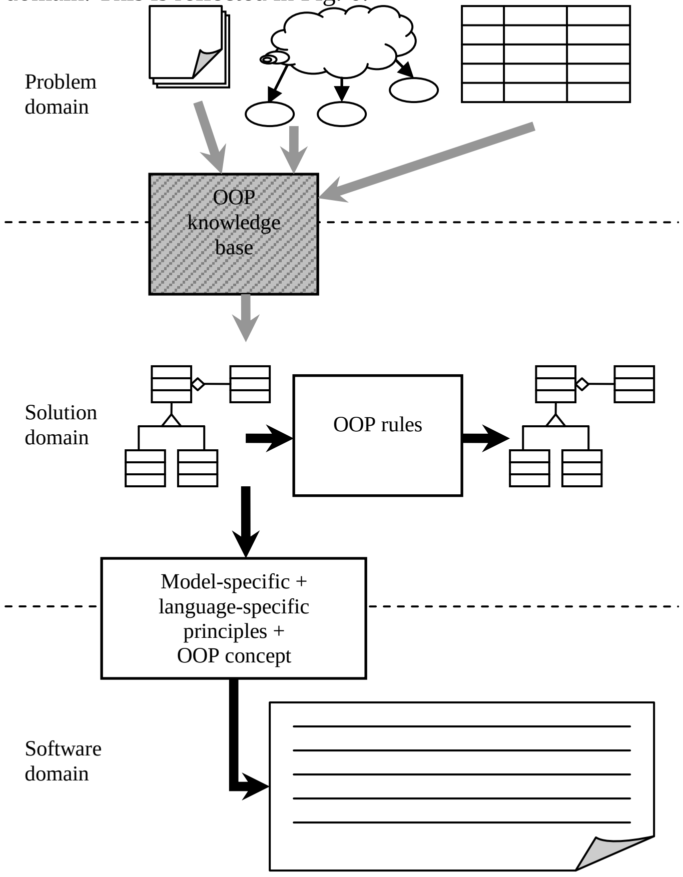


Figure 6. Relation of the knowledge-based code generator with sotware development domains

Theoretically, the OOP knowledge base can be used to transform some artifacts from the Problem software development domain into the model. However, such transformation is out of the knowledge-based generator's scope and thus, it will not be described in this paper.

The main disadvantage of the knowledge-based architecture is a significant amount of the work required to build a knowledge base and map its concepts with the syntax and XMI. However, after this job is done, the knowledge-based code generator potentially can be more powerful. The other disadvantage is that there is a significant amount of different ways to organize knowledge base as well as some variants to write a syntax templates, which means that functionality of such code generator can strongly vary depending on specialists and many other factors.

In addition, the proposed architecture can be used in two different dimensions: vertical (to generate a code from a model) and horizontal (to verify if model is correct). Normally, generating the code, both dimensions should be involved, but their separate usage is also possible depending on the task. When the model is only verified, the code generator uses mostly the rules from OOP knowledge base, but while performing only the code generation, all other parts of the proposed architecture are used. Fig. 6 shows how these dimensions are related with the software development domain. The reason of calling these two concerns as dimensions is also reflected there. Models are at the same level of abstraction – solution domain, so, while validating them, the code generator is staying within its bounds. That is why the dimension is horizontal. As for the vertical dimension, code generation transfers the model between the different states of the various domains – vertically.

IV. USAGE EXAMPLES OF THE KNOWLEDGE-BASED ARCHITECTURE

As it was mentioned before the architecture of the knowledge-based code generator can be used in two different dimensions: horizontal (to verify the correctness of the model) and vertical (to perform the code generation). The subsections below show the examples of both dimensions.

A. *Vertical Dimension (Code Generation)*

The knowledge-based code generator works with the OOP knowledge base in the first place. It iterates through the defined concepts starting from the root of the structural tree by jumping between elements according to the relations of these concepts. First, code generator takes an appropriate mapping from the model-specific principles and tries to find a value according to this mapping inside the XML meta-data. If the value is found, then, the code generator takes a syntax template for the OOP concept currently being iterated and produces an output. If the template interpreter finds any markup (text enclosed in "<>") then, it refers to the appropriate concept from the OOP knowledge base, searches for the values according markup from the model-specific principles and finds another template of the text to produce. When the code generator meets a structure enclosed in "*{ }*" it assumes that the model could contain none or more than one element that is represented by the markup inside. Therefore, it takes each of them, repeating the text and iterating through every other concepts enclosed in figure brackets as much as model elements it had found. If the code generator meets something inside "*[]*" then it produces an appropriate text if it finds any values inside the XML documents, otherwise it does not. If the code generator does not find any model elements which are enclosed in "*{}*" or "*[]*" brackets, it will not produce any text inside of them.

Concerning the example shown in Fig. 2, Fig. 4 and Fig. 5, the root is "*OOP*" and its children are "*Class*" and "*Interface*". The code generator will not find anything connected with "*Interface*" because XML document does not contain anything about it. But since a markup of interface is included inside the square as well as figure brackets, the code generator will not insert anything at the place of markup "<*Interface*>", as well as it will not produce a text "*implements*" and ",". The situation with the concept "*Class*" is different. Let us assume that this concept has a markup "*//model/class*". The code generator will use it to state that the XML document contains two elements expressed with this path so it will iterate through them. First of all, the code generator will produce the text "*class* " and meet the markup "<*Name*>". The knowledge base describes the concept with the same name, so the code generator will jump to a model-specific principle and find a markup for this concept. Let us say it is "*<element>/@name*". As the parent concept of the current one is Class, the full path to determine its name is "*//model/class/@name*". Using this, the code generator finds out the name of the class and produces the following code *"A {"*. After that it will return to the parent (which is the the concept Class) and continue parsing the template. The next stop will be a markup "<*Attribute*>". Here, in the same way, the code generator will take a visibility, type and the name of the attribute and construct a piece of code "*private int A_atr1;"*. Since no more information about the class A is provided the code generator will iterate further producing a text "class *B {}* ".

At first glance this mechanism is very similar to the ordinary templates, but the difference is that template is fully separated from the markup. A markup for the Class could possibly be "*//diagram/elements/class*" but for its name – "*//diagram/attributes[@id = <element>/@id]/name*". This never affects the template and vice versa because these two blocks are connected through the knowledge base which is static. That gives an opportunity to switch between markups easily without making any changes inside the templates.

B. *Horizontal Dimension (Model Verification)*

The rules which are used to validate the model are described in Fig. 3. The mechanism of the model verification is conceptually simple: the model's every element is tested on matching the defined rules and if at least one of them does not match, the model is considered incorrect. Despite appearing primitive in theory, this part of the proposed architecture is both the most creative and complex because the rules can be translated into logical expressions in a variety of ways. Each rule contains standard symbols defined by predicate logic [12] (terms, predicates, and, or, not, etc.), as well as references to the concepts from the OOP knowledge base expressed as variables. But in contrast to the model-specific and language-specific principles not every OOP concept must be described in the rules. The other part which is skipped in this example is putting some sense in predicates or, in other words, explaining to a computer what does they mean. The programming language, such as Prolog [17] is used to accomplish this. Although it does not fully feat in the concept of the knowledge-based architecture as well as in the code generation itself, it is specially created to work with logical expressions.

V. Related Work

The first code generators in the World were related to text-to-text transformation. They were nothing but high-level compilers. According to [4], the first scientist who started to talk about the code generation was Wilcox. In 1971, he described his compiler, which was based on two internal forms: Abstract Program Tree and Source Language Machine. The first one was translated into the second one, which in turn was transformed into the machine code.

The first popular code generator which was able to transform model into a source code was Rational Rose [18] developed by Rational Software in 1997. Later, this company was consumed by IBM resulting with evolution of Rational Rose into Rational Software Architect [19]. The tool's integration into an Eclipse environment allows users to customize their transformations more flexibly. Flexibility is a distinctive feature of Eclipse, so some other tools operating under this platform exist: Acceleo [20] and XPand [21]. The other popular tools – the "monsters" of today's industry which provide a code generation opportunities are such tools as SPARX Enterprise Architect [11] and Microsoft Visual Studio [22]. This list could be populated with a significant amount of other smaller tools, and basically, all of these code generators use their own different transformation mechanisms which are mostly based on templates. In addition, none of these tools are positioned to use artificial intelligence to perform code generation.

The template based programming originated in the 1960s and became especially popular thirty years later [23]. Eighteen years later, in 2008, the template-based code generation approach was also standardized by OMG [24]. However, since then, no new versions of this specification appeared.

The idea of using artificial intelligence in the field of the code generation was expressed by bloggers-enthusiasts as well as by scientists. Danilchenko and Fox [23] describe their system called the Automated Coder using Artificial Intelligence (ACAI), which as they claim is "… *a first pass at a purely automated code generation system*". ACAI generates the code through some simple steps: first, it generates a plan(s) to solve the problem; next, it takes reusable code components from the library and weaves them according to a created plan. The result is a text template which has been processed to get a working source code. ACAI uses an artificial intelligence technique called Case-Based Reasoning which can be used to maintain a reusable library of code components. Case-Based Reasoning is popular, and also is used in the other code generation systems: CHEF [25], Software Architecture Materialization Explorer [26] and The Individual Code Reuse Tool [27].

The knowledge-based code generator studies, which are mentioned above are advanced and actually they are far from the classic MDA concept. The studies are based on building the program's text from the reusable code components. The knowledge-based architecture, however, describes more simple mechanism which uses only basic AI principles but in fact is much similar to the ideology of the Model-Driven Architecture.

VI. Conclusion

Abstraction is the process by which we extract and distill core principles from a set of facts or statements. A model is an abstraction of something in the real world, representing a particular set of properties. There are two primary reasons developers build the model [28]: understanding a process or a thing by identifying and explaining its key characteristics and documenting ideas what developers need to remember and to communicate those ideas to other. OMG's last initiative – Model Driven Architecture offers the third reason on using the models during software development [29]. Using models as a basis for the further code generation and UML class diagram plays the central role on moving an idea about the code generation into the industry.

A significant amount of different standards in the code generation area overwhelmed it and as a result, led to the lack of ways of using them in practice. However, a significant amount of tools exist that have an ability to generate a more or less working source code. In general, all of them are using templates as a code generation technique, and this could be a reason why those code generators have not got an ability to work perfectly yet. The main problem is that templates do not provide any mechanism to verify a model which could be wrong from the start. Thus, as long as completely new approaches of code generation will not be found, the idea of using MDA for making the process of implementing fully functioning system more easy, affordable and reliable will remain nothing but a utopia. For now, templates could not be fully replaced, that is why they must be used in conjunction with the other methods.

The authors of this paper wanted to make a computer "smarter" for the code generation tasks. This could be achieved by applying some principles of the artificial intelligence. Therefore, authors propose a knowledge-based architecture which separates a code generator into three main blocks: model-specific, language-specific principles, and OOP knowledge base. The first one is used to perform meta-model mapping, the second one describes the syntax of a programming language, and the third one keeps the main principles of OOP, as well as it serves as a bridge between the first and the second block. In the opposition to the simple template, the proposed architecture keeps the meta-model mapping independent from templates. It allows not only to use different syntax with different mapping cases but also involving different specialists to work with them independently in turn to save the time.

The key contribution of this paper is extending an ideology of the MDA central components, such as templates, meta-model and constraints. According to the architecture proposed by authors, the templates are no longer overwhelmed by complex directives but contain only references to the OOP knowledge base – the names of OOP concepts. They also represent not only concrete code mapping situations, but a whole syntax of the particular programming language. The templates are independent from the XMI mapping rules because of the OOP knowledge base which is restricted by the first order logic rules that are an alternative to MDA OCL. In contrast of this language, the predicate rules are also independent from any concrete syntax and XMI, as well as they describe global OOP constraints based on the knowledge base. In addition, the described architecture's components do not only reflect the basic MDA components, but also represent the basic AI structures, which means that they have a potential for future studies of making code generator cleverer.

The code generator, which is based on such an architecture, can be used not only to perform the code generation, but also to verify the model. The both tasks could be performed separately as well as together. The knowledge-based code generator has a potential ability to become powerful, however it is very important to make a good OOP knowledge base.

The further researches will be connected with adding details to each of the three described levels: finding better structures to express them, forming some restrictions and formal rules for this task. When the concept of the knowledge-based architecture is fully ready, the tool should be implemented to realize it practically. This tool could be used to validate the presented approach by systematically applying some tests, which display the most problematic aspects of the model to code transformations, including those which other tools can not handle.

ACKNOWLEDGEMENTS

The research presented in the paper is partly supported by Grant of Latvian Council of Science No. 342/2012 "Development of Models and Methods Based on Distributed Artificial Intelligence, Knowledge Management and Advanced Web Technologies".

REFERENCES

[1] Object Management Group, [Online]. Available: www.omg.org [retrieved: September, 2013]

[2] Model Driven Architecture FAQ, [Online]. Available: http://www.omg.org/mda/faq_mda.htm [retrieved: September, 2013]

[3] UML Unified Modeling Language Specification, OMG document, [Online]. Available: http://www.omg.org/spec/UML/2.4.1 [retrieved: September, 2013]

[4] R. G. G. Cattell, A survey and critique of some models of code generation. Tech. rep. Pittsburgh, Pennsylvania, USA: School of Computer Science, Carnegie Mellon University, 1979.

[5] J. Sejans and O. Nikiforova, "Practical Experiments with Code Generation from the UML Class Diagram," Proceedings of MDA&MDSD 2011, 3rd International Workshop on Model Driven Architecture and Modeling Driven Software Development In conjinction with the 6th International Conference on Evaluation of Novel Approaches to Software Engineering, Osis J., Nikiforova O. (Eds.), Beijing, China, SciTePress, Portugal, Printed in China, Jun. 2011, pp. 57-67

[6] T. Stahl and M. Volter, Model-Driven Software Development, Wiley, 2006, pp. 428.

[7] I. Jacobson, G. Booch, and J. Rumbaugh: The Unified Software Development Process, Addison-Wesley, 2002, pp. 512.

[8] O. Nikiforova, A. Cernickins, and N. Pavlova, "Discussing the Difference between Model-driven Architecture and Model-driven Development in the Context of Supporting Tools," Proceedings of the 4th International Conference on Software Engineering Advances, IEEE Computer Society, Sept. 2009, pp. 446-451.

[9] OMG: Catalog Of OMG Modeling And Metadata Specifications, [Online]. Available: http://www.omg.org/technology/documents/modeling_spec_catalog.htm [retrieved: September, 2013]

[10] A. Bajovs, Research of the Basic Principles of the Model-To-Code Transformation, Bachelor Thesis, Riga Technical University, 2012.

[11] Enterprise Architect – UML Design Tools and UML CASE Tools for Software Development, [Online]. Available: http://www.sparxsystems.com.au/products/ea/index.html [retrieved: September, 2013]

[12] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Prentice Hall, 2002, pp. 1132.

[13] M. Marcotty and H. Ledgard, The World of Programming Languages, Springer, 1986, pp. 380.

[14] OMG: MOF 2 XMI Mapping Specification Version 2.4.1, [Online]. Available: http://www.omg.org/spec/XMI/2.4.1 [retrieved: September, 2013].

[15] XML Path Language (XPath) 2.0 (Second Edition), W3C document, [Online]. Available: http://www.w3.org/TR/xpath20 [retrieved: September, 2013]

[16] A. Cernickins, O. Nikiforova, K. Ozols, and J. Sejans. "An Outline of Conceptual Framework for Certification of MDA Tools," Proceedings of the 2nd International Workshop on Model-Driven Architecture and Modeling Theory-Driven Development, In conjunction with ENASE 2010, In Janis Osis, Oksana Nikiforova, (Eds.), Athens, Greece, SciTePress, Jul. 2010, pp. 60-69.

[17] C. S. Mellish and W. F. Clocksin, Programming in Prolog: Using the ISO Standard, Fifth Edition, Springer, 2003, pp. 300.

[18] IBM Software – Rational Rose, [Online]. Available: http://www-01.ibm.com/software/awdtools/developer/rose [retrieved: September, 2013]

[19] Introducing IBM Rational Software Architect, [Online]. Available: http://www.ibm.com/developerworks/rational/library/05/kunal/?S_TACT=105AGX99&S_CMP=CP [retrieved: September, 2013]

[20] Acceleo home page, [Online]. Available: http://www.eclipse.org/acceleo/ [retrieved: September, 2013]

[21] XPand – Eclipsepedia, [Online]. Available: http://wiki.eclipse.org/Xpand [retrieved: September, 2013]

[22] Microsoft Visual Studio 2012, [Online]. Available: http://www.microsoft.com/visualstudio/eng/team-foundation-service [retrieved: September, 2013]

[23] Y. Danilchenko and R. Fox, "Automated Code Generation Using Case-Based Reasoning, Routine Design and Template-Based Programming," in the Proceedings of the 23rd Midwest Artificial Intelligence and Cognitive Science Conference, S. Visa, A. Inoue and A. Ralescu editors, Omnipress, Apr. 2012, pp. 119-125.

[24] MOF Model To Text Transformation Language, Version 1.0, [Online]. Available: http://www.omg.org/spec/MOFM2T/ [retrieved: September, 2013]

[25] K. J. Hammond, "CHEF: A Model of Case-based Planning," in Proceedings of the Fifth National Conference on Artificial Intelligence, AAAI, Aug. 1986, pp. 267-271.

[26] G. Vazquez, J. Pace, and M. Campo, "A Case-based Reasoning Approach for Materializing Software Architectures onto Object-oriented Designs," in Proceeding SAC '08 Proceedings of the 2008 ACM symposium on Applied Computing, ACM, Mar. 2008, pp. 842-843.

[27] M. Hsieh, and E. Tempero, "Supporting Software Reuse by the Individual Programmer," in Proceedings of the 29th Australasian Computer Science Con"ference, Australian Computer Society, Inc, Jan. 2006, pp. 25-33.

[28] J., W. Satzinger, R. B. Jackson, and S. D. Burd: Object-Oriented Analysis and Design with the Unified Process, Thomson Course Technology, 2005, pp. 656.

[29] D. Gasevic, D. Djuric, and V. Devedzic: Model Driven Engineering and Ontology Development, 2nd edition, Springer, 2009, pp. 378.

Towards a Smart City Security Model

Exploring Smart Cities Elements Based on Nowadays Solutions

Felipe Silva Ferraz, Carlos Candido Barros Sampaio,
Carlos André Guimarães Ferraz, Gustavo Henrique da Silva Alexandre, Ana Catharina Lima de Carvalho

Informatics Center
Federal University of Pernambuco
Recife, Brazil
{fsf3, ccbs, cagf, ghsa}@cin.ufpe.br

CESAR – Recife Center for Advanced Studies and Systems
Centro de Estudos e Sistemas Avançados do Recife
Recife, Brazil
{fsf, ccbs, ghsa, aclc}@cesar.org.br

Abstract— **Even though concepts related to smart cities are well established and spread, those concepts are still very thin when related to Information Security. This paper will present some studies on smart cities, and will show that those studies are based on three macro concepts, System Interoperability, Applications and Frameworks/Platforms. Solutions and tools focusing on Information Security are still far from the common and typical scenario of urban systems. Based on that assumption, we propose a solution, based on a self-contained information security model, that aims to relate several items from urban system sand solutions for problems like, privacy and information integrity. This paper presents the first stage of this model that is based on elements found on nowadays solutions form Smart Cities.**

Keywords-smart city; security; privacy; information.

I. Introduction

The term City, in general, means a place or an urban area demographically closed, running under economical and political understanding [1]. These assumptions are also related to the idea that a City is a trading and commercial center that offers different services and products to a region. Those images are directly influenced by the industrial age when the production of services and products had their transformation [1].

Today, urbanization have reached an unprecedented level; different from other ages, large cities have now most part of world population and an increasingly share of the world's most skilled, educated, creative and entrepreneurial minds [2]. More than 50 percent of people on the planet now live in large cities [4]. According to United Nations, this number will increase to 70 percent in less than 50 fifty years [3]-[5]. This so-called *city growth* or emerging of urban life is driving the city infrastructure into a stress level never seen before as the demand for basic services are both increased and overloaded [6].

According to a research called *Smarter Cities and Their Innovation Challenges*, there is an urgent need for urban scenarios and cities to be smarter in the management of their infrastructure resources and interactions [3]. The urban performance must not depend only on its hardware infrastructure, or the physical concepts of infrastructure, but it must start taking into account social interactions and a faster deployment of information and services.

Cities are becoming increasingly empowered technologically as their core systems, i.e., Education, Public Safety, Transportation, Energy and Water, Healthcare and Services, are instrumented and interconnected, enabling new ways to deal with massive, parallel and concurrent usage.

In this paper, we aim to present a Security Model for Smart Cities, based on the assumption that this field has few works focusing on Information Security and its consequences. To that, this paper will present works related to security, it will depict Smart City initiatives and will present the Security Model based on urban system, data type and their interaction. This work is divided as follows:

In Section II, we present a difference between information security in cloud computing areas and smart cities. In Section III, there is going to be a detailed model basis. Section IV presents model's entities explanation; Section V presents the conclusion and future works.

II. Information Security, Cloud Computing Security and Smart Cities

Typically, cities or urban areas will begin to increase the demand for a better and more spread network connectivity, which will serve as a base for a group of different and more powerful features and services. Along with that, potential threats against those systems will increase, going beyond security network aspects. Hence, security measures will be needed within system scope. According to Bartoli et al. [7] and to Li et al. [8], for an effective protection of a Smart City system or its environment in the correct way, a number of problems related to security have to be addressed following a specific plan, definition or architecture. Those plans cover different types of systems and threats, but still do not address specific environmental situations and entities of a Smart City.

Although systems information security, within the scope of smart city, is not a well-established concept, another area presents several advances in this security field; for instance, CERT presents a hierarchical graphic where it presents potential vulnerabilities and/or exploits to be studied as challenges in Cloud Computing area [9]. G-Cloud Information [10], on the other hand, presents a series of minimum requirement needed by Cloud Computer Service Provides (CCSP).

Finally, in Security Architectures for Cloud Computing [11], international trends in security requirements for Cloud computing, along with security architectures proposed by Fujitsu such as access protocol, authentication and identity (ID) management, and security visualization, is presented and discussed.

Different from the smart city needs, Cloud Computing studies on security focus on specific problems for this area, among those problems we can mention topics like virtualization, PaaS (Platform as a Service) or IaaS (Infrastructure as a Service) or SaaS (Software as a Service) failures, legal responsibility, scalability to ensure availability. Even though smart city systems rely on Cloud Computing as a host and service provider for its services, it is still a scenario where security concerns supersedes far beyond the structure it uses.

As a basic situation, we can explore an application that helps citizen to report crimes; this typical application is deployed within a cloud computer structure to guarantee scalability and availability among others. Still, even a secure cloud solution does not create the guarantee for the citizen, or user, that its identity will be kept private in case of a complaint; nevertheless it also does not answer the question *"Is this accusation, reliable? Should the system trust this complaint? Should it relay on historical denounces to trust this one?*"

This situation summarizes a common concern with privacy within the use of a Smart City system. Many more can be presented, like a patient who does not want his/hers medical history reviled, but still, they must be accessible to the medical entities. Another situation is of a driver that would not want his physical location broadcasted but has to have its location ready available for the traffic authorities in case of a traffic transgression or does NOT need its location just to inform cases of traffic violation. Whatever the situation, the smart city system(s) presents different needs from a Cloud Computing system, for instance, because it needs to deal with a higher, and therefore different, level of concern.

III. BACKGROUND

A century ago, city population would not exceed the size of a million people. Nowadays that scenario is known in more than 450 cities [4]. The connection from the services and structures of those cities has become a big connected information system in order to guarantee that the cities are becoming smarter and, from that, will endure as a Smart City, and not just a connected city.

Within these scenarios, Smart City environments, or solutions, we face three specific topics that are: System Interoperability, Platforms and Applications.

A. System Interoperability

In the last decades, major cities around the globe have emerged to a reality in which every major public and urban system are now represented in the form of a Computer System. Urban systems like the ones responsible for Education, Public Safety, Transportation, Energy and Water, Healthcare and Services are now present and vital to the continuity of those cities. Furthermore, those systems deal with a historical amount of data that would be impossible to manage in any different way.

One of the problems faced by those environments is that their solutions are isolated from each other, therefore it is impossible to gather information from one system and use it on another system so that it creates more valuable information [2]-[6], [12]. To face that, research studies show that is vital for cities, which want to have a smarter and healthier growth, to open their system to make possible for other entities to interact with as many system as possible to provide to the citizens, public and private institutions with more valuable information [3], [4], [6].

B. Platforms or Frameworks

Once it is understood that urban system face problems related to their interconnection, a second approach lays on the proposal or the creation of platforms or frameworks to connect different units, to interact through this platform. Those units are represented in the form of a set of specific profiles that are directly related to citizens, buildings or companies and Things [13]-[17].

In this option, there is a highly adopted concept of The Internet of Things [18]-[21] which create situations where sensors and different entities can and will interact with each other. Furthermore we have the concept of social sensors, which are presented by values provided directly by citizens through social networks like, Twitter or Facebook. Even though social networks are a well established concept representing an important step to reduce distances and connecting people, its importance to urban life lies upon the messages, or posts, created by the user (citizen) itself [18], [22], [23], which leads to a vision that one citizen, or its information, is equally important as any other citizen. This way, Platforms and Frameworks emerge as the infrastructure in which the concept of sensor information, which could be either physical or a social sensor, is used as input to instantiate specific solutions for different urban environment. For instance we have Cosm [19], former Pachube, a platform for Energy connection that uses a physical sensor to monitor energy consumption on Twitter profiles that tracks traffic problems, working as a social sensor.

C. Applications

The important difference between those two topics (Platform x Applications) is that a platform is built with the assumption that the power to decide how it is going to be used depends upon the choices made by the user that instantiate it. For instance, it is possible to see the same platform built to serve as a dynamic panel showing opinions or as a medical solution showing the status of all systems in a hospital [21]. Hence, we are dealing with an approach more abstract, which usually comes combined with an application as a solution.

On the other hand, solutions made for urban systems that are represented by applications appear as more dedicated, practical and less abstract. Some relevant examples are Waze

[24] and Catch the Bus [25] which are applications that show problems related to traffic, Dwolla [26] that attacks scenarios of economical behavior and Crime Reports [27] for security measures, and even a Big Data based localization system called SkyBox [28] that aims to, through satellite photos, make easier and faster localization in different environments.

IV. SECURITY MODEL FOR SMART CITIES

To represent our security model, from a previous analysis on several platforms, applications and interoperable solutions focusing on smart city situations, we summarized entities that somehow are presented in every one of the analyzed subjects. Those entities are: System Type, Sensors, Actuators, Sensitivity Level and Grouping Value.

The following sections will explain one by one the selected entities and what they represent.

A. System Type

In a given system type we exhibit different types of system that are involved in Smart City areas. They are: Education, Public Safety, Transportation, Energy and Water, Healthcare and Government Services.

Education Systems: Represent every system that is, directly or indirectly, related to educational services.

Public Safety: Represent every system that aims to help public areas and citizens to guarantee city safety such as, but not only, vigilance systems or crime reports systems.

Transportation System: Represents every system that, in different way, drives citizens into a better movement around a city. The movement could be either with or without using automotive transportations.

Energy and Water System: Defines as any system that acts directly focusing on natural resources, more specifically on Energy or Water.

Healthcare System: Every system that seeks to improve the health and well-being of a patient.

Government Services: This term depicts every system that works within government scenarios. It can vary from a justice web system that expose legal issues of each citizen, to a platform that opens governmental data to the city itself. For example, we have the *Open Government Data* and British *Data.gov.uk*, both under this same idea [29], [30].

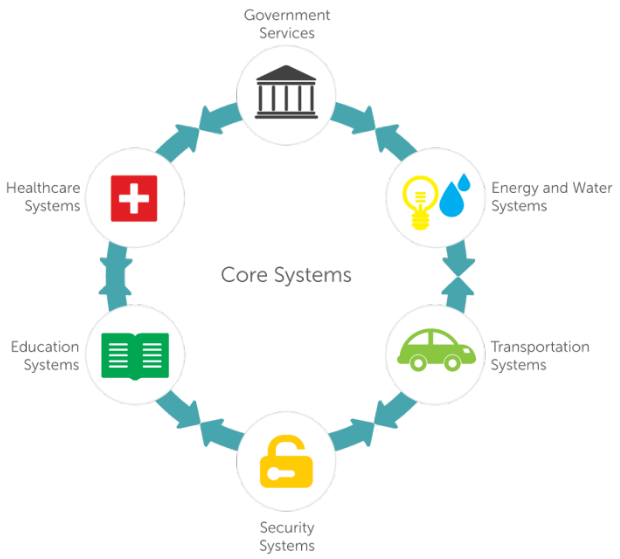


Figure 1. Core Systems Relations.

Figure 1 illustrates an environment that contains all systems mentioned above. Even though the image presents some relations between some systems, it is important to state that this is a common example and not the mandatory communication.

B. Sensors

In Sections A and B, we presented different types of sensors that are part of a Platform or Application. Both areas work with the same concept. In those areas we find entities responsible for gathering information. In our model, those entities express themselves as Physical Sensors and/or Social Sensors.

Sensors that generate an expected format of data and non-personal information represent a Physical Sensor i.e., Thermal Sensors, Presence Sensors, Magnetic Sensors, Radio Frequency Identification (RFID) tags and others.

A Social Sensor represents an entity whose data is created from a person and contains personal information, for example, a post on Twitter or any other social network.

C. Actuators

A sensor, Physical or Social, represents entities responsible for gathering information from the environment. Hence, an actuator represents the ways that the information gathered by the Sensor layer, processed or not, is sent back to the user. Take for example a system that collects information about traffic, combining twitter with physical traffic sensors, and sends back to the driver's Smartphone information about which part of city present more or less traffic. This way, both the application and the Smartphone are Actuators.

The actuators can be one of, Direct or Indirect; This classification will depend on the access to the information. The access can be direct, like on a Smartphone or indirect, like through a smart panel.

D. *Sensitivy Level*

In this section, we present a situation on which information is gathered from different types of sensors and could be delivered back by different actuators. In this particular scenario, the collected data is used as grouped information. Grouping value will be the last entity we will approach, mainly because it needs more than just one value to represent a correct data. Once this requirement is fulfilled, it is necessary to take under consideration that this one value is a sensible value, and could not compromise the identity of the citizen that, through the sensor, sent its location and traffic info. Based on that, the information has to respect the Sensitive Level that can be private or public.

Private information cannot be exposed, even further, it cannot be associated with its creator. On the other hand, Public information can be associated, expose and even stored for future use.

E. *Grouping value*

The last aspect is grouping value and it represents one of three possible states any information represents. Those three states are, Information Grouped, Information Non-Grouped and Reversible Information Grouped.

Information Grouped represent a group of information that does not make sense if analyzed or stored individually, e.g., numbers, values or medians.

Non-Grouped Information represents all kinds of information that have value if analyzed or stored individually, e.g, Dates, Coordinates or social posts.

All information that represents a value when presented in a grouped fashion, but that can be traced back to its individual values, are called Reversible Grouped Information. An example of this concept would be List of values or Map Areas.

V CONCLUSION

The presented entities that compose the model are summarized in Table 1.

TABLE I: ENTITIES SUMMARIZED

Entity	Classification	Description
System type	Education Public Safety Transportation Energy and Water Healthcare Government	References to the systems type that is related to.
Sensors	Physical Social	Refers to the mechanims used to gather information from the citizen.
Actuators	Direct Indirect	Related to the way in which the information is returned to the user.
Sensitivity Level	Private Public	Depicts about the level that the information can be used.
Grouping Level	Grouped Non-Grouped Reversiable	Deals with the value degree that an information has according to its grouping need.

This paper has proposed a first stage of a Security Model that aims to add more Information Security to Smart Cities solutions. The elements presented in Table I are entities suggestion based on studies and analysis made on some of the solutions listed in this work. Furthermore as future works, we intend to develop the second stage of the presented model. It will present, a relation between the entities presented here with which aspect that is more critical in terms of information security. As a final work we suggest to present architectural solutions based on Security Pattern, those will guide city administration towards more secure urban systems.

REFERENCES

[1] R. J. Johnston and D. Gregory, The Dictionary of Human Geography. Blackwell Reference, 1981.

[2] D. Susanne, G. Constantin, & K. Mary. Smarter Cities for Smarter Growth: How Cities "Can Optimize Their Systems for the Talent-Based Economy". Somers, 2010, NY: IBM Global Business Services.

[3] M. Naphade, G. Banavar, C. Harrison, J. Paraszczak and R. Morris, "Smarter Cities and Their Innovation Challenges," Computer, vol.44, no.6, pp.32,39, June 2011.

[4] Dirks, S., Keeling, M., and Dencik, J., " A Vision of Smarter Cities: How Cities Can Lead the Way into a Prosperous and Sustainable Future. Somers, NY: IBM Global Business Services, 2009.

[5] D. Washburn and U. Sindhu, “Helping CIOs Understand ‘ Smart City ’ Initiatives”, forrester, 2010.

[6] A. Caragliu, C. Del Bo, and P. Nijkamp, “Smart Cities in Europe”, Journal of Urban Technology, vol. 18, no. 2, Apr. 2011, pp. 65–82.

[7] A. Bartoli, M. Soriano, J. Hernandez-Serrano, M. Dohler, A. Kountouris, D. Barthel, Security and Privacy in your Smart City , in Proceedings of Barcelona Smart Cities Congress 2011, 29-2 December 2011, Barcelona (Spain).

[8] W. Li, J. Chao, and Z. Ping, “Security Structure Study of City Management Platform Based on Cloud Computing under the Conception of Smart City,” 2012 Fourth International Conference on Multimedia Information Networking and Security, Nov. 2012, pp. 91–94.

[9] W. R. Claycomb and A. Nicoll, "Insider Threats to Cloud Computing: Directions for New Research Challenges," Computer Software and Applications Conference (COMPSAC), 2012 IEEE 36th Annual , vol., no., pp.387,394, 16-20 July 2012.

[10] HM Government, “G-Cloud Information Assurance Requirements and Guidance,” 2012, avaliable at http://gcloud.civilservice.gov.uk/files/2012/05/G-Cloud-Services-IA-Requirements-and-Guidance-version-1-0-_for-publication_1-2.pdf.

[11] M. Okuhara, “Security Architectures for Cloud Computing,” pp. 397–402, fujitso (2010).

[12] Accenture, Cisco and the GSMA, Smart Mobile Cities: Opportunities for Mobile Operators to Deliver Intelligent Cities Contents, 2011, available at: http://www.accenture.com/SiteCollectionDocuments/PDF/Accenture-Smart-Mobile-Cities.pdf.

[13] A. Attwood, M. Merabti, P. Fergus, and O. Abuelmaatti, “SCCIR: Smart Cities Critical Infrastructure Response Framework,” 2011 Developments in E-systems Engineering, Dec. 2011, pp. 460–464.

[14] L. Lugaric, G. S. Member, S. Krajcar, and Z. Simic, “Smart city — Platform for emergent phenomena power system testbed simulator," Innovative Smart Grid Technologies Conference Europe (ISGT Europe), 2010 IEEE PES , vol., no., pp.1,7, 11-13 Oct. 2010

[15] H. Chourabi, T. Nam, S. Walker, J. R. Gil-Garcia, S. Mellouli, K. Nahon, T. a. Pardo, and H. J. Scholl, “Understanding Smart Cities: An Integrative Framework,” 2012 45th Hawaii International Conference on System Sciences, Jan. 2012, pp. 2289–2297.

[16] M. Al-Hader, A. Rodzi, A. R. Sharif, and N. Ahmad, “Smart City Components Architicture,” 2009 International Conference on

Computational Intelligence, Modelling and Simulation, 2009, pp. 93–97.

[17] F. Gil-Castineira, E. Costa-Montenegro, F. Gonzalez-Castano, C. López-Bravo, T. Ojala, and R. Bose, “Experiences inside the Ubiquitous Oulu Smart City,” Computer, vol. 44, no. 6, Jun. 2011, pp. 48–55.

[18] D. J. Skiba, “The Internet of Things (IoT).,” Nursing education perspectives, vol. 34, no. 1, 2011, pp. 63–4.

[19] “Cosm.” [Online]. Available: https://cosm.com. [Accessed: 10-Aug-2013].

[20] “Sensetecnic.” [Online]. Available: http://sensetecnic.com/. [Accessed: 10-Aug -2013].

[21] M. Blackstock, N. Kaviani, R. Lea, and A. Friday, “MAGIC Broker 2: An open and extensible platform for the Internet of Things,” 2010 Internet of Things (IOT), Nov. 2010, pp. 1–8.

[22] E. Duravkin, "Using SOA for development of information system “Smart city”," Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), 2010 International Conference on , vol., no., pp.258,258, 23-27 Feb. 2010

[23] J. Guevara, E. Vargas, F. Barrero, S. Member, and S. Toral, "Ubiquitous architecture for environmental sensor networks in road traffic applications," Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on , vol., no., pp.1227,1232, 19-22 Sept. 2010

[24] “Waze.” [Online]. Available: www.waze.com. [Accessed: 10-Aug -2013].

[25] “Catch the bus.” [Online]. Available: http://catchthebusapp.com/. [Accessed: 10-Aug -2013].

[26] “Dwolla.” [Online]. Available: www.dwolla.com. [Accessed: 10-Aug-2013].

[27] “Crime Reports.” [Online]. Available: https://www.crimereports.com/home/iphone_app. [Accessed: 10-Aug-2013].

[28] “Skybox Imaging.” [Online]. Available: http://www.skyboximaging.com/. [Accessed: 10-Aug-2013].

[29] “Open Government Data” [Online]. Available: http://opengovernmentdata.org. [Accessed: 10-Aug -2013].

[30] “Data.gov.uk” [Online]. Available: http://data.gov.uk/. [Accessed: 10-Aug-2013].

Camera Trajectory Evaluation in Computer Graphics Based on Logarithmic Interpolation

Mikael Fridenfalk
Department of Game Design
Uppsala University Campus Gotland
Visby, Sweden
mikael.fridenfalk@speldesign.uu.se

***Abstract*—A new technique is presented within the field of multimedia software applications, based on a logarithmic shape-preserving piecewise cubic Hermite interpolant for evaluation of camera trajectories in mathematically generated large-scale geometries, such as 3D fractals, with the ability to eliminate the oscillations that currently are associated with interpolation of exponential zooms.**

***Keywords*-fractal space; logarithmic; LPCHIP; PCHIP; spline**

I. Introduction

Piecewise cubic Hermite splines are presently used for high-end interpolation of the trajectories of cameras and 3D objects in computer graphics [2,6,7], such as computer games, but also for computer-controlled cameras in film production.

On an implementation level, the standard method to control a camera in computer graphics is by an object called the target camera [8], defined by camera position, a look-at position and the orientation of the camera around the vector pointing from the position of the camera to the look-at position (called roll). To avoid causing the viewer disorientation or nausea, roll is often set to a constant value.

Presently, the spline interpolation techniques that are used in computer graphics are as a rule not based on shape-preserving ones, here defined as interpolants that are both harmonic and monotonic, such as the Piecewise Cubic Hermite Interpolating Polynomial (PCHIP) in MATLAB [1,4,5], which could be a better choice for camera trajectory control, since PCHIP eliminates the overshooting effects that are associated with the regular variant, see Figure 1 (left), thereby increasing the level of control in camera trajectory design without any practical downside, see Figure 1 (right). A reason for this could be that MATLAB, which is the application that introduced PCHIP to a wider audience, is presently not widely used in systems for generation of motion picture, but rather applications such as image processing.

In Figures 1-2 (left), the trajectories of two sets of breakpoints are evaluated by a regular piecewise cubic Hermite interpolant. While the implementation of the harmonic mean in Figure 1 (middle), eliminates the overshooting effects of the regular interpolant, Figure 2 shows that the harmonic mean does not always work properly, unless the tangents (or slopes) m_1 and m_2 are limited by locally monotonic constraints, see Figures 1-2 (right).

The main difference between a regular and a shape-preserving piecewise cubic Hermite interpolant is that here, the tangents m_1 and m_2 in the regular interpolant are functions of the mean values of the differences of adjacent *breakpoints* (or *keyframes*), while the shape-preserving version is based on locally monotonic functions of the harmonic mean of the same, see LPCHIP (for Logarithmic Piecewise Cubic Hermite Interpolation Polynomial) in Figure 12 for the example that was used for the generation of the graphs in this paper. LPCHIP in a non-logarithmic mode (*i.e.*, with the argument *lg* set to *false*), henceforth called the PCHIP equivalent, is not identical to the MATLAB function PCHIP, but a simplified version. The principal difference is that the PCHIP equivalent is designed specifically for a constant step size between the breakpoints. However, by the addition of separate interpolation along the horizontal axis, as shown in Figures 1-2, the step size between the breakpoints becomes automatically variable.

In Figure 3 (left), the effect of camera trajectory evaluation is demonstrated using the regular mean value for the evaluation of m_1 and m_2 in LPCHIP (with *mode* set to REGULAR) and in Figure 3 (middle), with the adjustments of m_1 and m_2 by multiplication with a factor of 0.25 instead of 0.5. As shown, while in the latter figure the overshooting effect of the camera position trajectory is reduced compared with the former, at the same time the look-at position trajectory has become rougher. This issue may be addressed by adaptive control, but is by default solved by the application of the PCHIP equivalent, see Figure 3 (right).

This paper consists of the presentation of a new technique and a comparison with standard techniques presently used in computer graphics, represented by the term *regular interpolation*. In Section 2, the application of a logarithm is studied in context with camera trajectory interpolation in exponential zooms, to eliminate the oscillating effects that were discovered using standard interpolation. In Section 3, a detailed solution to the oscillation problem is offered, including the evaluation of interpolation points as a function of arbitrary points in time. This solution was further visually verified by implementation in a computer graphics application primarily designed for visualization of 3D fractals.

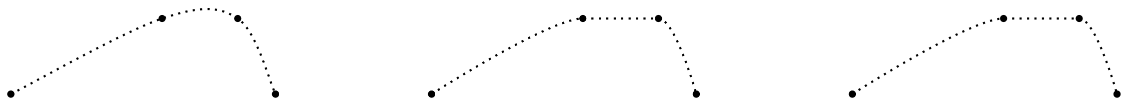


Figure 1: Regular (left), harmonic (middle), harmonic and monotonic (right). As shown in this example, the regular interpolant causes a slight overshoot between the second and the third breakpoints.

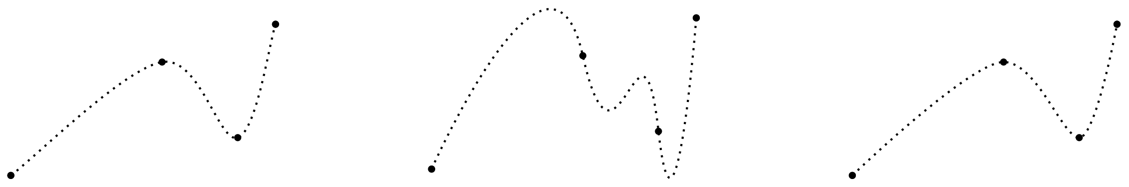


Figure 2: Regular (left), harmonic (middle), harmonic and monotonic (right). While harmonic interpolation solves the overshooting problem of the example in the previous figure, to work properly for all cases, it has to be monotonic.

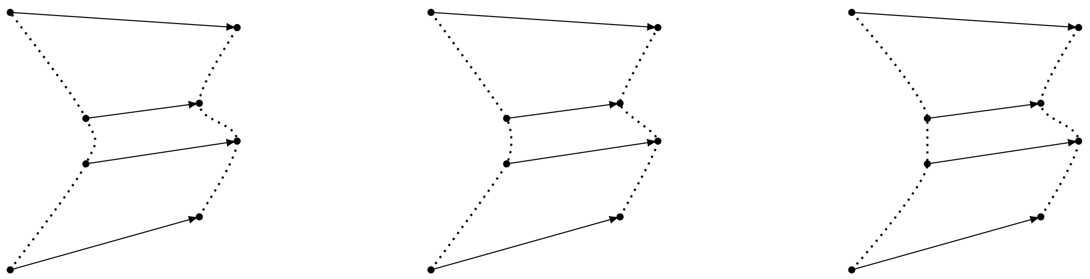


Figure 3: Regular (left), regular with adjusted weights (middle) and the PCHIP equivalent (right).

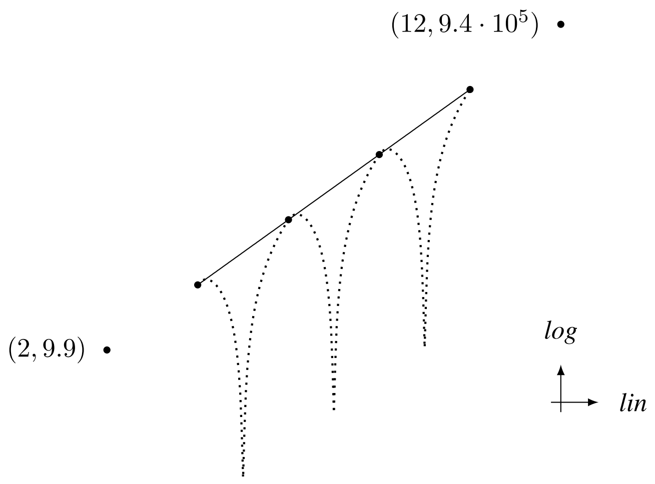


Figure 4: Regular interpolation (dotted) versus LPCHIP (solid) in an even exponential zoom.

II. Logarithmic Extension

We developed a camera trajectory control system using Apple Xcode [9], based on a PCHIP equivalent during the NASA International Space Apps Challenge 2013, for the production of a video within the Ad Infinitum project on the challenge *Why We Explore*. Although the control system worked perfectly well within local room dimensions, yet the exponential zoom from microcosm to macrocosm showed to work less than satisfactory due to an uneven change of the experienced zooming speed.

This effect is demonstrated in Figures 4-8 by the dotted curves. In Figure 4, the effect is best shown using a regular cubic Hermite interpolant with six breakpoints defined by the function 3.146^x, which was the largest base with three decimals that could be used before the interpolant caused a singularity. In this context, x represents the linear horizontal axis in Figures 4-10. As shown in Figures 5-8 (dotted curves),

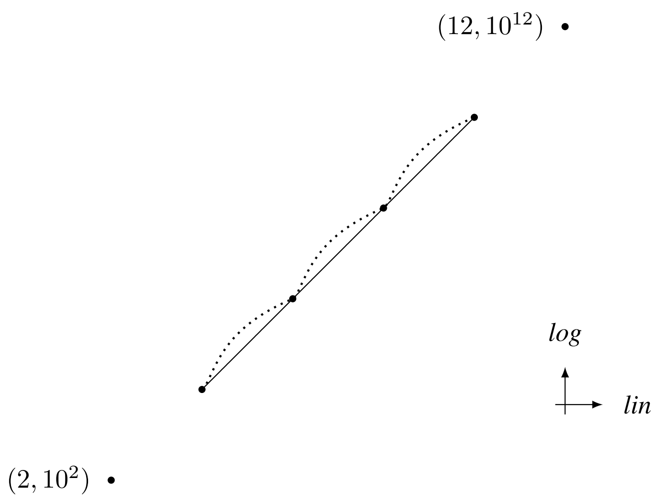


Figure 5: The PCHIP equivalent (dotted) versus LPCHIP (solid) in an even exponential zoom.

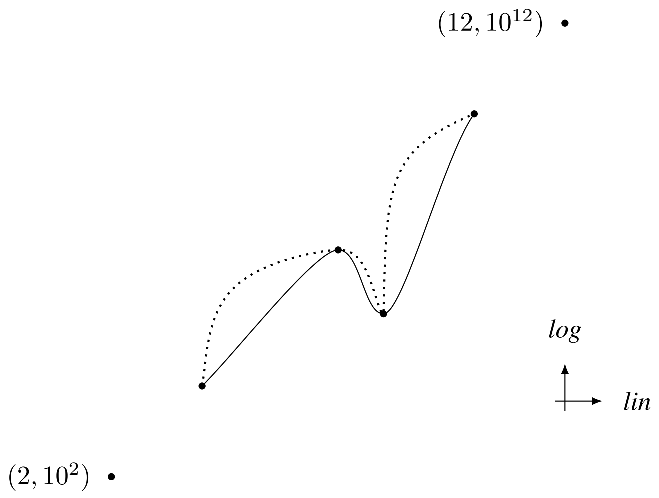


Figure 6: The PCHIP equivalent (dotted) versus LPCHIP (solid) in a dynamic exponential zoom.

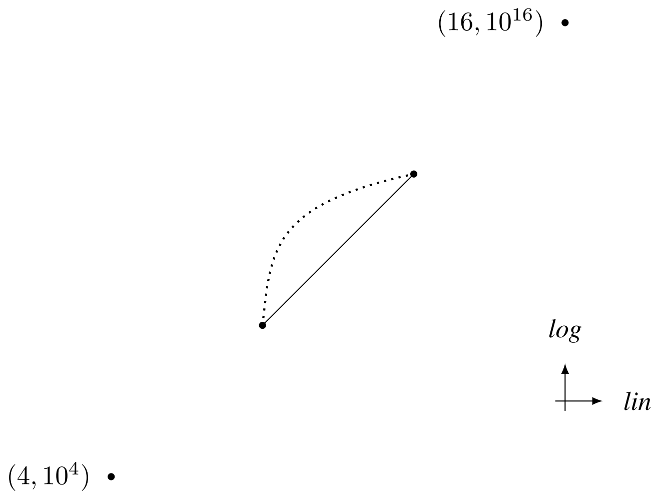


Figure 7: The PCHIP equivalent (dotted) versus LPCHIP (solid) in a moderately scaled exponential zoom.

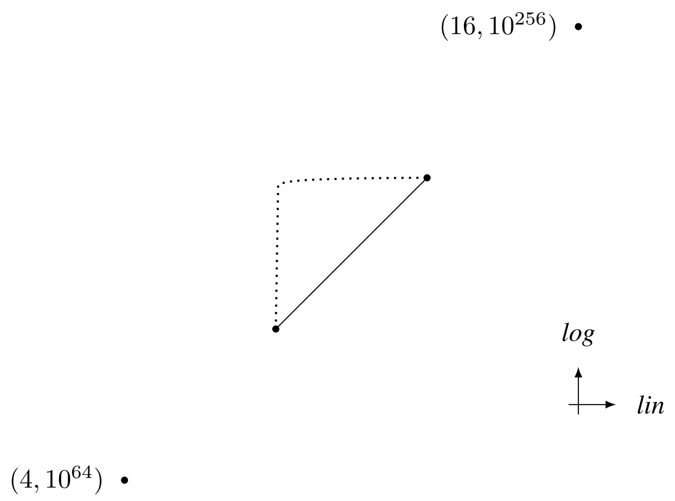


Figure 8: The PCHIP equivalent (dotted) versus LPCHIP (solid) in a large-scale exponential zoom.

where Figure 8 displays a zoom using the function 10^{16x}, the PCHIP equivalent did not cause any singularities. In this context, it is possible to increase the robustness of the regular interpolant by weight adjustments, although as previously demonstrated at a tangible cost. The conclusion from Figures 7-8 is that it is possible to minimize the oscillation effects of PCHIP (and its equivalent) in exponential camera zooms, if the breakpoints are placed closely enough. The problem is however that the whole idea using interpolation is to eliminate the manual generation of the finer details of a trajectory. Thus, PCHIP (and its equivalent) fails to operate properly if the distance between the breakpoints are exponentially increased.

III. RESULTS

The solution to this problem showed to be that the interpolant that was implemented in the Ad Infinitum project, had to be redefined to be able to map the distance between the look-at and the camera position into logarithmic space (and back after the interpolation was performed), which is done by setting the LPCHIP argument *lg* to *true* in Figure 12. Thus, the solid lines in Figures 4-8 are obtained, which are identical to the desired trajectory we initially wanted the regular and the PCHIP equivalent interpolants to follow.

This new interpolant is called Logarithmic PCHIP (LPCHIP), a name inspired by the MATLAB PCHIP function. However, in order to work properly, the new interpolant has to be implemented with some caution, since any position value equal or less than zero exceeds the range of the function.

The solution is therefore not to apply LPCHIP (in logarithmic mode) directly on camera trajectory breakpoints, one for each of the six dimensions (three degrees of freedom for the camera position and three for the look-at position), but rather only to interpolate the distance between the camera and the look-at position, since by definition, this distance can never be equal or less than zero. Thus, the arguments x_0 to x_3 of LPCHIP in Figure 12 do not have to be limited by any safeguards.

Cam_LPCHIP in Figure 13 shows how the new technique is implemented in practice. Briefly expressed, the interpolation is performed the conventional way by separation of the mentioned six degrees of freedom. However, the difference here is that using LPCHIP, the distance between the look-at point and the camera is modified so that it follows a logarithmic trajectory instead of a Euclidean.

In the sample code in C++ that is presented in Figures 12-15 (which in this specific case was assessed to be as clear and succinct as pseudocode for this level of detail, but more straightforward to implement), mCamCoords is a matrix of the type *double* of size mCamCoordsN $\times$ 7, where each row consists of a breakpoint and the first column consists of the time associated with each breakpoint followed by the camera position (columns 2-4) and the look-at position (columns 5-7).

A question in this context is how LPCHIP affects interpolation where the distance between the breakpoints are relatively constant (or more specifically non-exponential). Figure 9 shows that the deviation between the PCHIP equivalent and LPCHIP is in this specific example too small to be visually detectable in this graph. In Figure 10, the difference between the PCHIP equivalent and LPCHIP (in Figure 9) has been magnified, which for this example gives a peak and mean deviation equal to 0.0043 versus 0.0011. This is relatively insignificant and hardly even noticeable for camera trajectory control applications, since the deviation is a smooth curve without any discontinuities.

This example is however only a near best case and in real applications the deviation should be usually quite visible. As an example, in Figure 11, the corresponding average deviation was estimated to 0.14 (or 2.4%), which is a more realistic number. A large number, such as this, is however not necessarily a disadvantage for LPCHIP compared with the PCHIP equivalent but could rather be a measure of the discrepancy of the latter compared with a well-designed interpolant specifically developed for camera trajectory evaluation.

Regarding the evaluation of t in Cam_Auto in Figure 14, to be accurate, a reverse interpolant has to be used. Although such interpolant can be derived symbolically, the solution showed to be relatively complex. This is why a more pragmatic approach was adopted by the application of Newton's method [3], where h'_k for any k denotes the derivative of h_k in:

$$t_{i+1} = t_i - \frac{h_1 x_1 + h_2 x_2 + h_3 m_1 + h_4 m_2 - y}{h'_1 x_1 + h'_2 x_2 + h'_3 m_1 + h'_4 m_2} \qquad (1)$$

The InvPCHIP method in Cam_Auto in Figure 14, takes y as an argument and returns t. This method is obtained by the addition of (1) inside a loop after the evaluation of h_k and h'_k in the PCHIP equivalent (with an iteration start value of $t_0 = 0.5$). Figure 11 shows an example of the application of LPCHIP as a function of time, using inverse time interpolation to obtain a smooth trajectory based on six breakpoints using identical start and end-points with totally 500 interpolation line segments. In the example in Figure 11, it took in average 4.08 versus 4.72 iterations to find a solution within an error interval in Newton's method of 10^{-6} versus 10^{-9}. In this case, when the time is measured in seconds, this is equal to accuracy levels in the order of microseconds versus nanoseconds.

Note that for correct performance, the current implementation of this camera trajectory evaluation technique requires a continuously increasing time value along the first column of mCamCoords.

IV. Conclusion

The new camera control system suggested in this paper showed to exceed current systems used in computer graphics. This new system is categorized by (1) utilization of a local monotonic function of the harmonic mean for the evaluation of the tangents of the piecewise cubic Hermite interpolator (in similarity with PCHIP), in combination with (2) operation in logarithmic space instead of Euclidean regarding the evaluation of the distance between the camera and the look-at point, thereby eliminating trajectory oscillations associated with interpolation of exponential zooms using present techniques.

Acknowledgments

Many thanks to the NASA International Space Apps Challenge 2013, where the mathematical problem solved in this paper was first identified. In addition, special thanks to Ellinor Dahl at Almi Gotland and Mikael von Dorrien at Almi Väst for the encouragement that led to this work.

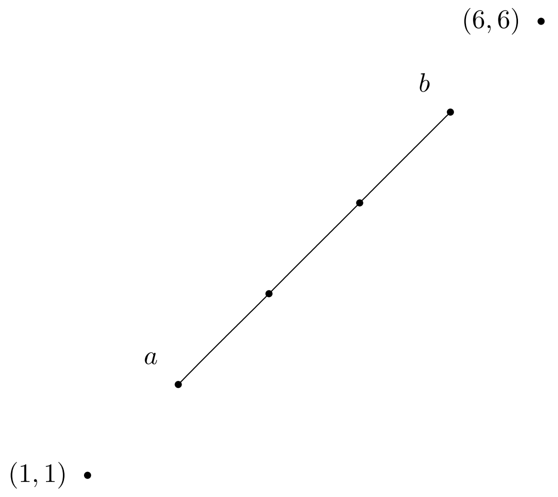


Figure 9: In a non-exponential zoom, the PCHIP equivalent and LPCHIP virtually coincide in this example.

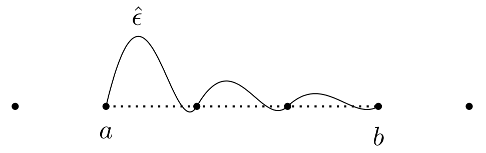


Figure 10: The difference between the PCHIP equivalent and LPCHIP in previous figure, gives a peak value of $\hat{\epsilon} = 0.0043$.

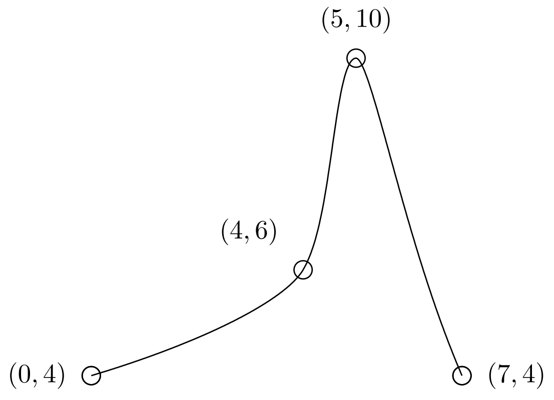


Figure 11: An example of LPCHIP interpolation as a function of time during 7 seconds. To obtain the correct parameter value t in Cam_Auto, an inverse version of the PCHIP equivalent is used.

References

[1] C. de Boor, K. Höllig, and M. Sabin, "High accuracy geometric Hermite interpolation", Computer Aided Geometric Design, vol. 4 (1987), no. 4, pp. 269-278.

[2] M. Christie, P. Olivier, and J. Normand, "Camera Control in Computer Graphics", Computer Graphics, vol. 27 (2008), no. 8, pp. 2197-2218.

[3] P. Deuflhard, Newton Methods for Nonlinear Problems: Affine Invariance and Adaptive Algorithms, Springer, 2011.

[4] F. N. Fritch and R. E. Carlson, "Monotone Piecewise Cubic Interpolation", SIAM Journal on Numerical Analysis, vol. 17 (1980), no. 2, pp. 238-246.

[5] C. Moler, Numerical Computing with MATLAB, Society for Industrial and Applied Mathematics, 2010.

[6] T. Mullen, Mastering Blender, John Wiley & Sons, 2010.

[7] T. Palamar and E. Keller, Mastering Autodesk Maya 2012, Sybex, Hoboken, NJ, USA, 2011.

[8] H. Smith, Foundation 3ds Max 8: Architectural Visualization, Dreamtech Press, 2007.

[9] Xcode, Apple Inc. <https://developer.apple.com/xcode/> [retrieved: August 7, 2013].

```
double GFX::LPCHIP(double x0, double x1, double x2, double x3,
                   double t, int mode, bool lg){

    if (lg){x0 = log(x0); x1 = log(x1); x2 = log(x2); x3 = log(x3);}

    double epsilon = 1e-20;
    double den, m1, m2, d0, d1, d2, t2, t3, h1, h2, h3, h4, y;
    d0 = x1 - x0; d1 = x2 - x1; d2 = x3 - x2;
    bool a0 = mode == SHAPE_PRES && (d0 * d1 < 0.);
    bool a1 = mode == SHAPE_PRES && (d1 * d2 < 0.);
    bool b = fabs(d1) < epsilon;

    if (mode >= HARMONIC){
        if (a0 || fabs(d0) < epsilon || b ||
            fabs(den = 1./d0 + 1./d1) < epsilon) m1 = 0.;
        else m1 = 2./den;
        if (a1 || b || fabs(d2) < epsilon ||
            fabs(den = 1./d1 + 1./d2) < epsilon) m2 = 0.;
        else m2 = 2./den;
    }
    else {m1 = .5 * (d0 + d1); m2 = .5 * (d1 + d2);}

    t2 = t * t; t3 = t2 * t;
    h1 =  2. * t3 - 3. * t2 + 1.;
    h2 = -2. * t3 + 3. * t2;
    h3 =       t3 - 2. * t2 + t;
    h4 =       t3 -      t2;
    y  = h1 * x1 + h2 * x2 + h3 * m1 + h4 * m2;

    if (lg) return exp(y); return y;
}
```

Figure 12: The LPCHIP interpolant (a pedagogic version), called by Cam_LPCHIP.

```
void GFX::Cam_LPCHIP(int idx, double t, bool lg){
    double X[6];
    For (i,6) X[i] = LPCHIP(mCamCoords[idx-1][i+1],
                            mCamCoords[idx][i+1],
                            mCamCoords[idx+1][i+1],
                            mCamCoords[idx+2][i+1],
                            t,SHAPE_PRES,false);
    if (lg){
        double dx[4],dy[4],dz[4],d[4],dist,eye[3],u[3],factor;
        For (i,4){
            dx[i] = mCamCoords[idx+i-1][1] - mCamCoords[idx+i-1][4];
            dy[i] = mCamCoords[idx+i-1][2] - mCamCoords[idx+i-1][5];
            dz[i] = mCamCoords[idx+i-1][3] - mCamCoords[idx+i-1][6];
        }
        For (i,4) d[i] = sqrt(dx[i]*dx[i]+dy[i]*dy[i]+dz[i]*dz[i]);
        dist = LPCHIP(d[0],d[1],d[2],d[3],t,SHAPE_PRES,true);
        For (i,3){eye[i] = X[i]; mLookAt[i] = X[i+3];}
        For (i,3) u[i] = eye[i] - mLookAt[i];
        factor = dist/sqrt(u[0]*u[0]+u[1]*u[1]+u[2]*u[2]);
        For (i,3) u[i] *= factor;
        For (i,3) mEye[i] = mLookAt[i] + u[i];
    }
    else For (i,3){mEye[i] = X[i]; mLookAt[i] = X[i+3];}
}
```

Figure 13: The LPCHIP camera control method, called by Cam_Auto.

```
void GFX::Cam_Auto(){
    int idx = mCamera_CurrentInterpIdx;
    if (mCamCoords[idx+1][0] < mTime && idx < mCamCoordsN-3)
        mCamera_CurrentInterpIdx = ++idx;
    double t = InvPCHIP(mCamCoords[idx-1][0],
                        mCamCoords[idx][0],
                        mCamCoords[idx+1][0],
                        mCamCoords[idx+2][0],
                        mTime,SHAPE_PRES);
    Cam_LPCHIP(idx,t,true);
    glLoadIdentity();
    gluLookAt(mEye[0], mEye[1], mEye[2],
              mLookAt[0], mLookAt[1], mLookAt[2], 0.0, 1.0, 0.0);
}
```

Figure 14: The main evaluation method, called once for each rendered frame.

```
#define For(i,N)  for (int (i) = 0; (i) < (N); (i)++)
...
class GFX ... {
    enum {REGULAR, HARMONIC, SHAPE_PRES};
    ...
    double mTime;//Time in Seconds
    double mLookAt[3], mEye[3];//Camera Position
    static const int MAX_CAM_COORDS_N = 1024;
    double mCamCoords[MAX_CAM_COORDS_N][7];//Breakpoints (Including Timestamps)
    int    mCamCoordsN;//The Total Number of Breakpoints
    int    mCamera_CurrentInterpIdx;//Current Breakpoint (Start Value = 1)
};
```

Figure 15: A selection of declarations.

Metaphors Applied to Interaction Design in Group Learning

Anderson Cavalcante Gonçalves, Deller James Ferreira
Informatics Institute
Federal University of Goiás
Goiânia, Brazil
andersongoncalves@inf.ufg.br, deller@inf.ufg.br

***Abstract*—The acquisition of the ability to use metaphors effectively contributes in increasing students' capacity to analyze and design interfaces. The use of metaphors in interaction design offers consistent interfaces, simple and intuitive. However, it is not easy for students to learn how to use metaphors in interaction design. To teach students how to develop interactive experiences through metaphors is not an easy task. This paper proposes a method for teaching the use of metaphors, while designing the website, desktop, mobile or tablet interface and presents the results of a case study on the successful teaching method proposed. The teaching method developed is a collaborative learning model based on model of King questioning and creative dimensions of Ferreira. It consists of creative tasks coupled with structured questionnaires with questions and are designed to encourage interaction, group learning, and foster creativity of students.**

Keywords-metaphors; interaction; design; teaching; learning

I. Introduction

Metaphors create connections between concepts that are already familiar to people. Metaphors explore the existing knowledge of each person to assimilate something new. Thus, the person is able to learn new things, using their previous knowledge of the world [1]. This means that the person will be able to understand and experience one kind of thing in terms of another [2]. Considering interaction design, it is desirable to provide an interface familiar to the user, easy to learn and use.

The use of metaphorical concepts is one of the resources available for creating intuitive user interfaces, simpler to learn and use. Entertainment websites, online stores, social networks, and others require an interface easy to learn and use. The interaction design should be well organized, easy to be interpreted and used by the users. Metaphorical concepts can be used in an expressive way to achieve this goal. Metaphorical concepts are pervasive in the culture of a society. Lakoff and Johnson [2] stated that metaphors are concepts inherent to subconscious and govern our whole way of thinking. Thus, the good use of metaphors in interface design is a feature that will make the interaction much easier to understand. Nielsen and Molich [3] established that we should minimize the cognitive load of the user. In other words, they stated that the designer should facilitate the reasoning required to interpret an interface. Also, they state that, in a user interface, there must be a match between the system and the real world. The designer should use phrases and concepts familiar to the user, rather than system-oriented terms.

The use of metaphors is a powerful resource that can be applied to achieve these heuristics. The appropriate application of metaphorical concepts turns an interface into a better interface. The interface design consists in defining how content is organized and presented to the user [4].

The consistent use of metaphors in the context Human Computer Interaction (HCI) helps to reduce the cognitive load necessary for understanding the functionality of a computational interface. Students´ understanding about a good usage of metaphors in HCI improves their ability to properly critique and design computer interfaces.

The use of metaphors is evident in many patterns and interaction interface designs. Some examples of the use of metahpors in HCI are evident at Apple's desktop, pattern wizard, canvas plus pallet pattern, menus, buttons, dashboards, carousel pattern, breadcrumbs pattern, and so on. But, how to apply metaphors in interaction design is nota easy to learn, the metaphors may have simple literal comparisons and complex connections [1]. In addition, there are misleading uses of metaphors. It is not simple for students to learn how to use metaphors in interaction design. To teach students to develop interactive experiences by means of metaphors it is not an easy task. Students need to understand user experiences, concerns, skills, interests and expectations and must develop the ability to create good designs based on user's knowledge.

Constructing effective metaphors is to some extent a complex skill because it depends on the creative ability of designers to see new analogies, in order to choose the right set of correspondences. These correspondences have to enhance some aspects and hide others, because metaphorical mediation carries elements of the concept that are consistent, but also inconsistent when using metaphors to comprehend one thing in terms of the other. For Schwartz and Fischer [5], metaphors highlights levels of complexity as well as the need for sufficient support to build complex understandings, but they do not easily capture the diversity of contexts that students might experience that could lead to the same abstraction. One of the reasons why metaphors can be difficult to learn and teach, it is because they have a high level of complexity. Another reason is that students may have different interpretations, which makes teaching difficult.

According to Hodges [6], if we examine the metaphor closely, their connotations are often the darkest when applied to teaching. Having a problem in one's research is motivating; having a problem in one's teaching is, well, a problem. In order to overcome the difficulty to teach metaphors in HCI, we suggest the teacher must apply a teaching method that encourages creativity and also criticism in interaction design.

In this work, we aim to awaken and stimulate the use of metaphors in teaching and learning interaction design with the aim to stimulate students' abilities to discern what is a good or bad design, allowing students to differentiate an interaction design that it is aesthetically good but possesses a bad functional design, and to propose new ideas and solutions.

In this work, we present an innovative teaching method to teach metaphors in human computer interaction design that fosters student's creativity and criticism. This method is based on collaborative learning and creative dimensions proposed by Ferreira [7] and the discussion method proposed by King [8], as shown in Fig. 1.

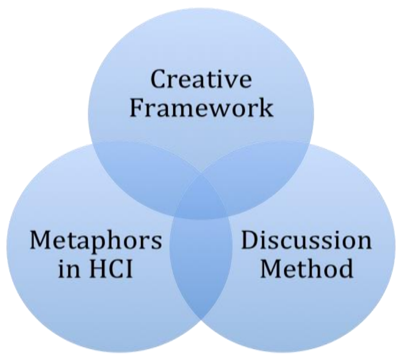


Figure 1. The proposed teaching method.

The creative dimensions, proposed by Ferreira [7], contain underlying dialogical processes that align dialogues with mental processes linked to both adaptive and innovative creativity. The creative dimensions constitute a pedagogical framework for designing exercises when teaching human computer interaction. They make it possible for teachers to create significant collaborative learning experiences to students, fostering them to activate mental processes underlying creativity during discussions.

On the other hand, in the discussion method group proposed by King [8], questions that trigger patterns of discourse in learning groups are designed to facilitate the construction of complex knowledge and problem solving.

Our teaching method proposes a combination Ferreira's framework [7] and King's [8] types of questions to propose a repertory of interaction design exercises exploring the use of metaphors. In our teaching method, we also approach the most common metaphorical concepts as structural, visual, functional, and positional metaphors, and consider where, when, why and how they are applied in the field of HCI. This metaphorical knowledge is part of the teaching method and is used during the tasks and questions created.

Their use can improve the computational interface and provide substantial gains in user productivity.

For example, visual metaphors are widely used in comic books. When a certain character is nervous, he is represented by a rough facial expression and smoke coming out of his ears, as shown in Fig. 2.

Figure 2. By means of prior knowledge acquired from the culture to which we operate, we recognize immediately that the character is nervous.

The Pinterest [9] website contains a virtual panel that makes possible to create image categories, including descriptions and comments. In this website, we have a visual metaphor that allows the user to act like in a real picture panel.

An example of positional metaphors application in interaction design is that the most important items must be at the top of the screen. This rule is very important in mobile applications.

In the LinuxMall [10] website, it is clear the use of functional metaphors. In this site, there is a backpack in the upper right corner, which suggests the user to place the desired products inside it.

As an example of the application of structural metaphors, commonly, an e-commerce website is subdivided into sections like in a real store.

In this work, we present a case study comparing the teaching method proposed in an undergraduate HCI class (treatment group) and a method involving students' discussions and informal teacher mediation in another undergraduate HCI class (control group). The case study conducted showed significant results.

In Section II, we show the importance of the application of metaphorical concepts to human computer interaction and show systematic aspects of metaphors. In Section III, we describe the teaching method proposed in this article. In Sections IV and V, we present a case study of the application of the teaching method proposed and the results obtained.

II. TEACHING METHOD FOR USE OF METAPHORS

The use of metaphors is essential for the user experience to become simple and intuitive. It facilitates user understanding and interactivity. According to Baumer [11], metaphors can be powerful aids for understanding because they can help the understanding of novel concepts.

However, learning to apply metaphors in computing environments is a difficult task. Although metaphors abound in human thinking, they can be surprisingly difficult to notice simply due to their ubiquity.

In this work, we developed a teaching method based on collaborative learning for teaching metaphors in interaction design. Collaborative learning is a successful method to awaken creativity. Creative solutions emerge from interactions that encourage students to express and evolve their ideas in specific problems.

According to Jonassen and Land [12], knowledge originates from productive discourse among individuals, the social relationships that bind them, and the physical artifacts, theories, models and methods that they use and produce. Productive discussions provide satisfactory results in collaborative learning, providing students the opportunity to share and co-construct knowledge.

Creative solutions are built during joint activities that trigger productive discussions. Creative and collaborative dimensions proposed by Ferreira [7] promote productive discussions, where students are encouraged to widen and deepen the design space. Students extend the design space when a new idea emerges and deepen the space of the project when an idea is developed.

Ferreira´s pedagogical framework allows the teacher to elaborate tasks that nourish creative discussions during collaborative problem solving in interaction design [7]. The author considers that creative products occur as stimulation of many different planes. The framework contains seven collaborative and creative dimensions to be applied by the teacher. According to Ferreira [7], the dimensions are: immersion, unpacking opportunities, exploring complementary ways, surpassing limits, expanding, discovering and developing unpredictable places. The dimensions contain dialogic processes that are dialogs aligned with mental creative processes associated to both adaptive and innovative creativity. Dialogic processes facilitate students to elaborate ideas built on other ideas, during their collaborations. The framework helps and challenges teachers to be aware of how complex students´ activities can be elaborated during collaborative learning. Considering students perspectives, during productive discussions they are able to detect relevant and irrelevant information, recognize the familiar, deal with new information, adapt and reapply techniques, among other creative important processes.

The use of provocative questions is another strategy that encourages students to interact productively. The students absorb and transcend knowledge when they engage themselves in profitable interactions.

King´s model approaches provocative questioning to induce relevant cognitive, meta-cognitive and socio-cognitive processes in participants [8]. Effective learning interactions induce complex cognitive processes including the analytical thinking necessary to create metaphors.

According to King [8], learning is constructed during interaction with others. During the interaction the students engage in the exchange of ideas, opinions and perspectives. The speech is composed of provocative questions, explanations, justifications, assumptions and conclusions. The construction of knowledge occurs when students explain concepts to each other. The questioning is a procedure that asks questions and answers. The interaction during the discussion results in a high level of learning. The model proposed by King consists of structured questions on issues of entry [8]. For example:

- How much similar to?
- How does it relate to?
- What do you remember and why?

Comprehension questions, for example:

- What does it mean?
- What's important for?

Connection questions, for example:

- How is similar with?
- What is different between?
- How can it be used for?
- What are the strengths and weaknesses?

The method proposed in this paper involves the development of group assignments focusing on the use of metaphors in HCI. Using our method, the teacher is able to elaborate group tasks and questions that encourage students engage themselves in productive discussions.

The teacher is invited to approach the dimensions proposed by Ferreira [7], questioning the model proposed by King [8] and knowledge about metaphors when designing exercises.

This way, the students have the opportunity to scrutinize metaphors in different contexts and are urged to find solutions and improvements in the application of metaphorical concepts in interaction design.

III. THE CASE STUDY

The case study aims to examine the effectiveness of teaching the use of metaphors in interaction design by means the proposed teaching method.

In this preliminary case study, four tasks based on the proposed method were analyzed. The preliminary results indicated that the proposed teaching method has potential to help teachers to mediate students' creativity when using metaphors in interaction design.

The students investigated were engaged in two classes of undergraduate Software Engineering at Federal University of Goiás in 2011 and 2012. There were 44 students in the class of 2011 and 42 students in the class of 2012. Each class was divided into groups of 6 (six) students and each group was evaluated by means of discourse analysis of online discussions.

A. Students' Profile and Communication Tools Used

Students are studying Software Engineering at the Federal University of Goiás. Students have the profile of software developers. They are learning about the concepts related to interface design, such as metaphors, usability guidelines and interaction patterns in the human computer interaction design course.

The communication tool used was the Moodle platform, which facilitates iterations among students. Each student posts messages concerning their responses and opinions. Moodle is a tool for course managing that can also be used for distance learning. Using the forums, the student can post a message at any time and place.

B. Description of Tasks (Treatment Group)

The tasks required are described following.

1) Discuss having in mind the questions related to the website Taisho [13].

Express your opinions and inferences, and propose appropriate solutions. In the following, we describe the questions regarding the Taisho website:

Why is it important to use a visual metaphor on the website? Are the elements observed on the website similar to real objects? How does the geisha and the shamisen relate to each other? Is the menu contained in the Website an example of positional metaphor? Did the visual metaphors facilitate user interaction in the website? Are the metaphors used readily apparent to any user? Why the metaphors were used? How each metaphor does interfere with the user's perception? How are the used metaphors similar to elements of everyday life? Are the metaphors used inherent in the culture of the target audience? What are the strengths and weaknesses of the use of visual metaphors in the website? Is the user able to associate the elements present in the metaphorical interface actions and objects represented? Does the website have a stable context? Does the positioning of the metaphors in the interface facilitate the identification of the company name? Does the name have a reasonable size and its location is noticeable? Are the different metaphorical elements in harmony? Do these elements contribute to the user understanding about the information contained in the website? Does the interface emphasize the services offered by the company? Are the interface services clear from the user perspective?

2) Discuss having in mind the questions related to the websites Sitotis [14] and Thedeepestsite [15].

During the discussions, you must engage critically and constructively with the ideas of others. Express your opinions, inferences, and propose appropriate solutions. In the following, we describe the questions regarding the websites:

The metaphor used in the logo of the company contributes to the understanding of company activity? Is it possible to satisfactorily answer the purpose of the website? Is the website interface sufficiently self-explanatory? Do the metaphors present in the website immediately contribute to the understanding of its interface? Do you understand the services offered by the website? Does he position of the website menu help the user to find the desired options in a simple and immediate manner? Can the user effortlessly navigate in the website? Is it able to distinguish the options? Is there a precise notion of what is in each option? Do the graphics and animations present on website show the actual content? Do the metaphors used emphasize a content or are merely illustrative? Is the user able to associate the elements present in the metaphorical interface actions and objects that they represent? Are the functional metaphors clearly perceived? Is the website interface sufficiently self-explanatory? Do the metaphors present in the website immediately contribute to the understanding of its interface? Can the user effortlessly navigate the website? Is the user able to distinguish the options?

3) Choose a Website to design your Mobile interface.

a) Take a look at the patterns shown in classes concerning mobile and navigation patterns. Also, take a look at the supplementary bibliography.

b) Use metaphors in the design of the website. Discuss having in mind the usability guidelines, particularly guidelines for mobile interfaces. Think outside the box when designing the website. Consider the following questions about metaphors:

- What types of metaphors are more suited to the context of your mobile interface?
- Do the metaphors used help the user to concentrate on the main service offered by the website?
- How visual metaphors can be used to enhance the understanding and simplicity of the website in a mobile environment without sacrificing your design?
- Is it possible to use metaphors to emphasize most relevant content to users?
- How can we subtly integrate metaphors and the graphic style of the website?
- Does the metaphors used provide users a logical path to follow, minimizing the effort required for understanding, making navigation easy and obvious?
- What functional metaphors can be used to facilitate the execution of some tasks?
- Can he use of metaphors make navigation easier and more intuitive for the user?

4) Each student must individually choose a context to adapt the wizard pattern using metaphors.

Defend your choice in your group grounding your arguments on the items "when" and "why" of the pattern. Each student must design a wizard and defend his idea, based on item "as" the wizard should be implemented. Discuss, choose and refine the best idea considering the in the following questions:

- Does the Wizard makes clear to the user what is the goal to be achieved?
- Is the user notified if he tries to start a new job before completing the current?
- Does he user have the option to go back and change the data entered in the previous step?
- It is visible to the user what is missing to achieve the goal?
- The Wizard is simple and intuitive and does not require much effort from the user understand how to use it?
- Do the metaphors used help the Wizard to became more simple and intuitive?
- Do the metaphors used help the user to concentrate on the goal to be achieved?
- Why metaphors were used? Do the metaphors significantly help the user reach success in every step and fulfill the purpose of the Wizard?

C. *Description of Tasks (Control Group)*

The tasks were in accordance to the following collaborative script:

Read a text about metaphors and evaluate the use of metaphors in the Websites Taisho and Sitotis. Based on the text and previous classes on this subject, express your opinions regarding the use of metaphors in the Website.

D. *Used in the Discourse Analysis*

The model used in the discourse analysis was proposed by Newman, Webb and Cochrane [16] and is described by ten categories:

1) *Relevance:* Relevant states or diversions.

2) *Importance:* Important points and issues or unimportant points and trivial issues.

3) *Novelty, new info, ideas, and solutions:* New problem-related information or repeating what has been said.

4) *Bringing outside knowledge or experience to bear on problem:* Drawing on personal experience or sticking to prejudice or assumptions.

5) *Ambiguities:* clarified or confused: Clear statements or confused statements.

6) *Linking ideas, interpretation:* Linking facts, ideas and notions or repeating information without making inferences or offering an interpretation.

7) *Justification:* Providing proof or examples or irrelevant or obscuring questions or examples.

8) *Critical assessment:* Critical assessment or evaluation of own or others' contribution or uncritical acceptance or unreasoned rejection.

9) *Practical utility (grounding):* Relate possible solutions to familiar situation or discuss in a vacuum.

10) *Width of understanding (complete picture):* Wide discussion or narrow discussion.

Categories 1 to 9 were explored in this case study.

E. *Model Used in the Creativity Analysis*

The model used in the analysis of creativity was proposed by Zeng, Salvendry and Zhang [17]. This model was structured in a checklist for web site design. The checklist comprises:

1) *Aesthetically appealing design:* artistic, colorful, energetic, beautiful, fascinating, entertaining, engaging, attractive, favorable, and desirable.

2) *Interactive design:* interactive, animated, available multimedia, and dynamic.

3) *Novel and flexible design:* unique, appealing, and flexible.

4) *Affective design:* stimulating, pleasing, delighting, and exciting.

5) *Important design:* relevant, important, and crucial.

6) *Common and simple design*: infrequent, unique and sophisticated.

7) *Personalized design:* personalized.

F. *Model Used in the Questions Analysis*

In the analysis of the questionnaire, were used dimensions of User Experience (UX) involving [18]:

1) *Immersion and Flow:* While the user is using the system he forgets everything around him.

2) *Tension:* The user feels tense while using the system.

3) *Competence:* The user thinks that he is good at using the system.

4) *Negative Affect:* The user feels bored while using the system.

5) *Positive Affect:* The user has fun while using the system.

6) *Challenge:* The user makes effort while using the system, but he takes pleasure in overcoming obstacles.

7) *Fellowship:* Good experiences are produced during social interactions.

8) *Discovery:* The user is pleased to learn new things.

9) *Expression:* The user is pleased to express new things and raises self-esteem.

G. *Results*

Each student was individually analyzed according to the model of Newman, Webb and Cochrane [16].

The result obtained by all students in the group, produced the group average. The average of all groups produced the overall result of the class.

Statistics of the overall outcome of the class in 2011 are shown in table I.

TABLE I. STATISTICS OF INTERACIONS IN 2011

Category	**Average**
1.Relevance	19.5%
2.Importance	18.5%
3.Novely, new info, ideas, solutions	3.25%
4.Bringing outside knowledge or experience to bear the problem	8.25%
5.Ambiguities	24.37%
6.Linking ideas, interpretation	9.37%
7.Justification	2%
8.Critical assessment	35.62%
9.Practical utility (grounding)	10.87%
Overall average considering all categories	14.63%

Each category was examined individually in each group and the results were obtained by calculating the percentage from 0 to 100 per category group. The percentage was obtained by examining the student's posts. Each student message posted was analyzed according to each category. The result was obtained by analyzing the positive factors of each category.

During the course in 2011, the teaching method proposed in this article was not used. The students were asked to evaluate and discuss the use of metaphors in websites considering no question.

Table II contains the general outcome of the interactions analysis in 2012.

TABLE II. STATISTICS OF INTERACTIONS IN 2012

Category	Average
1.Relevance	71.65%
2.Importance	58.73%
3.Novely, new info, ideas, solutions	32.86%
4.Bringing outside knowledge or experience to bear the problem	10.68%
5.Ambiguities	5.48%
6.Linking ideas, interpretation	10.27%
7.Justification	19.48%
8.Critical assessment	49.71%
9.Practical utility (grounding)	10.06%
Overall average considering all categories	29.82%

In category 1, we obtained 71.65% of relevant assertions. This result indicates that students had a significant improvement in the ability to make relevant statements. In category 2, it was obtained 58.73% of important issues. The result obtained in the category two indicates a significant improvement in addressing important issues. In the category 3, it was obtained 32.86% of new information, ideas and solutions. Students were able to propose new ideas, solutions and information. In category 4, we obtained 10.68%. Students were able to bring the information out of knowledge. In category 5, it was obtained 5.48% of ambiguities. In category 6, we obtained 10.27% of union ideas and new interpretations. In category 7, we obtained 19.48% of justification. Students were able to justify their ideas and affirmations. In category 8, we obtained 49.71% of critical assessment. The students' ability to make critical evaluations greatly improved. In category 9, we obtained 10.06% of practical utility. The average in all categories of the class of 2012 was 29.82%.

The results achieved were satisfactory. Compared with the class in 2011, class in 2012 achieved an overall gain of 15.19 percent. There was a clear improvement in all categories. In some categories there was a significant gain. Gains related to category 3 were 29.61 percent and earnings were related to category 8 of 14.09 percent. The category 1 and category 2 also greatly benefited. We note that the category 3 was the most favored. The students have acquired the ability to propose something new, new ideas and solutions, which is essential for a software engineer and interaction designer. Category 8, that is related to critical thinking, also had a great improvement.

The presented statistics show that the use of our teaching method in teaching the use of metaphors interaction design instigates and encourages any student to infer criticism and find more effective and creative solutions for the design of computational interfaces.

In tasks two and three we analyzed the products designed and presented by each group of students. In the analysis, we used the creativity checklist for website design of Zeng [12], analyzing important design factors, such as: aesthetically appealing design, interactive design, novel, and flexible design, affective design, design important, common and simple design and personalized design.

TABLE III. CREATIVITY CHECKLIST FOR WEBSITE DESIGN

Creativity Checklist	Classification
Aesthetically appealing design	Excellent
Interactive design	Good
Novel and flexible design	Good
Affective design	Good
Important design	Excellent
Common and simple design	Good
Personalized design	Excellent

The results were classified in excellent, good, regular or inappropriate. The analysis was based on the products presented by the students. Analysis of the products was successful. When making the checklist, we observed that the products obtained excellent results regarding the aesthetics. The interactive design achieved a good result. The novel and flexible design also achieved good results. The affective design which includes items such as stimulating and exciting achieved good results. The featured products have an important and relevant design; this result was excellent. In common and simple design products observed products with rare and sophisticated design. The result was classified as good. In personalized design, the result was excellent; all products owned a custom design. The results obtained in speech analysis have been confirmed in the analysis of the product. Students who possessed better performance in the categories of speech produced and presented the best products. The critics and creativity promoted by collaboration and productive interactions among students, trigged by the application of the teaching method, contributed effectively to student learning. The students applied the concepts discussed adequately. The discussions resulted in products of high quality design.

A questionnaire was developed with twelve questions to evaluate the experience gained by the students. Students who were involved in the groups that performed all the tasks proposed responded to the questionnaire. The analysis was performed according to the dimensions of UX [13].

TABLE IV. QUESTIONS ANALYSIS

Category	Average
1. Immersion and Flow	94.12%
2. Tension	85.3%
3. Competence	88.3%
4. Negative Affect	20.56%
5. Positive Affect	79.44%
6. Challenge	88.3%
7. Fellowship	50%
8. Discovery	85.3%
9. Expression	94.12%

According to the analysis 94.12% of the students forgot everything around them as they discussed the tasks. For 85.3% of the students the tension and difficulty decreased during task performance. 88.3% of students thought to be consistent inferring opinions. 79.44% of students felt excited. 88.3% of students felt challenged and encouraged to discover new ideas and solutions. 50% of students shared good experiences during social interactions. 85.3% of students felt happy to learn new things. 94.12% felt pleasure in expressing their ideas and had self-esteem by implementing these ideas. The results obtained were very satisfactory.

The results from Table II show that we have improved the results obtained on the control group by implementing our proposed method. The results obtained from Tables III and IV corroborate the results obtained from Table II, showing that the students pleasingly engaged in collaborative tasks and successfully developed creative products.

IV. CONCLUSION

It is not known in literature the existence of a teaching method to apply metaphors in interaction design. In this paper, we highlight the use of metaphors in interaction design. The use of metaphors improves the interaction design, providing more respect and importance to computational interfaces. We addressed different types of metaphors, such as visual, functional, structural, and positional metaphor. The proper use of metaphors produces a positive and significant impact on usability of user interfaces.

However, students find it difficult to learn and apply metaphorical concepts in interaction design. In order to overcome this problem, we addressed a teaching method to teach creativity and criticism in the context of interaction design using metaphors. A case study was designed and successfully applied. The preliminary results show that the teaching method based on collaborative learning through the development of questions that stimulate group discussion achieved good results. There was a significant improvement in the class where the method was applied compared to class where there was no application of the method.

This work contributes to teachers to arouse students' creativity, directing and encouraging them to infer creative solutions and to properly criticize interaction design. This contributes greatly to their learning. In this way, any student aggregates the knowledge necessary to criticize and design a more intuitive interface that is simpler to learn and use. All students tasks were contextualized in the use of metaphors in interaction design, as can be seen in the examples previously provided. Thus, both the discourse analysis and product analysis indicates that the use of metaphors was successful.

The results show the relevance of the study and the teaching method applied. However, more case studies are being performed as well as the discourse and products analysis are being done for more than one researcher to reduce the degree of subjectivity of the research.

ACKNOWLEDGMENT

This work is supported by FAPEG and CNPq – Brazil.

REFERENCES

[1] Dijk V., Betsy, Lingnau, A., Landoni, M., and Ruthwen I. Metaphorical Interaction Models/Interfaces. PuppyIR. University of Twente, July 2011.

[2] Lakoff, George and Johnson, Mark. *Metaphors We Live By*. Chicago: The University of Chicago Press, 2003.

[3] Nielsen, J. and Molich, R. Heuristic evaluation of user interfaces. Empowering People - Chi'90 Conference Proceedings. New York: ACMPress, 1990.

[4] Preece, J. Rogers, Y. and Sharp, H. Interaction Design: Beyond Human- Computer Interaction. New York, NY: John Wiley and Sons, 2007.

[5] Schwartz, M. S., and Fischer, K. W. Useful metaphors for tackling problems in teaching and learning. On Campus, 11(1), 2006, pp. 2-9.

[6] Hodges, Linda C. 2004. "The Problem as Metaphor in Teaching." The Nea Higher Education Journal, 2004, pp. 39-48.

[7] Ferreira, D.J. Human Computer Interaction Teaching Method to Encourage Creativity. Lisbon, Portugal: ICSEA, 2012, pp.472-478.

[8] O'Donnel, Angela M., and King, Alison. Cognitive Perspectives on Peer Learning. New Jersey: Lawrence Erlbaum Associates, 1999.

[9] Pinterest Website, https://www.pinterest.com/ (current Sep. 30, 2013).

[10] LinuxMall Website, http://www.linuxmall.com.br/ (current Sep.30, 2013).

[11] Baumer, S. P. E., Tomlinson, B, Richland E. L. and Hansen J. Fostering metaphorical creativity using computational metaphor identification. Proceedings of the seventh ACM conference on Creativity and cognition, New York, NY: ACM, 2009, pp. 315-324.

[12] Jonassen, D. H., Land, S. M. Theoretical foundations of learning environments. Preface. In D. H. Jonassen and S. M. Land (Eds.), New Jersey: Lawrence Erlbaum, 2000, pp. 3-9.

[13] Taisho Website, http://www.taishoflorianopolis.com.br/ (current Sep.30, 2013).

[14] Sitotis Website, http://www.sitotis.hr/ (current Sep.30, 2013).

[15] The Deepest Site, http://thedeepestsite.com/ (current Sep.30, 2013).

[16] Newman, D. R., Webb, B., and Cochrane, C. A content analysis method to measure critical thinking in face-to-face and computer supported group learning. St. Louis: University of Missouri–St. Louis, 1996.

[17] Zeng, L., Salvendy G., and Zhang M. January 2009. "Fator structure of web site creativity." Computer in Human Behavior. Vol. 25, pp. 568-577, January 2009.

[18] Poels, K., IJsselsteijn, W., and De Kort, Y. Development of the Kids Game Experience Questionnaire. Poster presented at the Meaningful Play Conference, East Lansing, USA, abstract in proceedings, 2008.

ProDec: a Serious Game for Software Project Management Training

Alejandro Calderón, Mercedes Ruiz
Department of Computer Science and Engineering
University of Cádiz
Cádiz, SPAIN
e-mail: alejandro.calderonsanchez@alum.uca.es, mercedes.ruiz@uca.es

Abstract – **Although there are some works related to the application of serious games for software project management training, there is a lack of tools that combine training and assessment in a single tool and that provide an environment for the learner where they can experiment decision making in real-life like scenarios. Project Decision (ProDec) is a simulation-based serious game created with the intention to train and assess students in software project management. The main objective is to take advantage of the engaging nature of games to place the learners in a virtual organization where they can manage software projects and solve real-life problems in a risk-free environment. For the trainer, ProDec is a support tool for training in matters such as leadership, task and team management, project monitoring and control, and risk management. It also helps the trainer assess the skills that the learners develop by playing the game. After any game play, ProDec offers a complete report including the logs representing every decision the players made and the result of applying the assessment criteria provided by the trainer at the beginning of the game play.**

Keywords - software project management; serious games; simulation

I. INTRODUCTION

Nowadays, the importance of teaching project management in computing curricula is out of discussion. In fact, the joint curricula developed by IEEE and ACM for Computer Science (CS), Computer Engineering (CE), Information Technology (IT), Information Science (IS) and Software Engineering (SE), currently under revision and planned to be released during the summer of 2013, acknowledge that computer professionals need training in project management. And not any kind of training, but a training that is beyond technical skills so that the future professionals develop professional practice during their studies.

Despite the importance that these curricula give to this topic and the increasing demand of the software companies seeking for professionals highly qualified in project management, very often we find that software project management syllabus are highly theoretical and quite uninteresting for the future professionals [1].

Compared with other studies, such as medicine, aeronautics, or engineering, computing future professionals do not receive the same practical training regarding real-life scenarios and rely on solving highly conceptual problems. As a consequence, novel professionals develop their experience working in real projects, where the effects of a wrong plan or decision-making can lead to a failed project or the loss of benefit for the companies they work for.

A serious game is a game designed for a primary purpose other than pure entertainment. Although serious games can be entertaining, their main purpose is to train or educate users. Based on this feature, this paper introduces ProDec, a serious game for software project management that helps:

a) Learners to develop and acquire practical experience in software project management, by allowing the players to plan a project, simulate its execution, track its performance and make decisions to keep the project on track.

b) Trainers to design real-world scenarios for developing learners' problem solving skills, and assess their learning.

c) Overcome the problems of lack of motivation of learners towards project management related subjects.

The structure of the paper is as follows: Section II shows the works related to our proposal; Section III describes the developed serious game, and Section IV shows how this game helps perform the learner's assessment. Finally, our conclusions and further work are given in Section V.

II. RELATED WORK

There exist numerous works related to the application of serious games for software engineering education. Most of these works have been retrieved and analyzed by Caulfield, Xia, Veal, and Maj in their systematic review of the literature [2]. However, if we focus on the field of software project management, the works found are scarce and quite specific. Within this area, the following tools are outstanding: SIMSOFT [3], SimSE [4] and DELIVER! [5].

SIMSOFT [3] is a serious game materialized as a printed game board, that shows the players the flow of the game, and a Java-based board, where the players can see the current and historical state of the project and adjust the project's settings. SIMSOFT mainly focuses on human resource management, with an emphasis on how the ability of the staff affects the outcomes of the project.

DELIVER! [5] is also based on a printed game board designed to help students develop the skills needed to measure and control project performance by applying the Earned Value Management technique. As stated by its authors, DELIVER! is mainly a game to motivate students in their learning process.

On the other hand, SimSE [4] is a serious game completely developed as a software tool that is based on software project simulation. SimSE allows students to practice a "virtual" software engineering process (or sub-process) in a fully graphical, interactive, and fun setting in which direct, graphical feedback enables them to learn the complex cause and effect relationships underlying the processes of software engineering. The game supports several development methodologies and focuses on the development of abilities for software process management.

Similar to SimSE, we can find SESAM [6], another serious game that uses a software application and simulation techniques to motivate learners in learning software project management. SESAM has a natural language interface and, during the game, records information about the game’s progress with the goal of showing several statistics at the end of the game.

If we focus on the educational objectives that can be achieved by using these games and compare them with a well-known taxonomy of learning objectives such as Bloom’s taxonomy [7], [8], we can find out that only SIMSOFT reaches the higher levels of the taxonomy, while the other tools place their educational objectives at the basic levels of the hierarchy, mainly the Knowledge level.

Regarding learners’ experience, the games already mentioned have been assessed through surveys so that the players provide some information regarding their experience where playing the game. However, the assessment of the learners’ new abilities developed by playing the games is always made by traditional methods and does not have any connection or feedback from the exercise of playing the game itself.

Unlike the above tools, ProDec does provide support for the learners’ assessment, by accepting and applying the assessment criteria that the instructor provides to the game tool.

III. Description of ProDec

ProDec is a serious game to teach software project management. The game is intended to be used at the end of an undergraduate course on software project management in computer science, information technology, information systems or software engineering programs. ProDec is intended to be a collaborative game, that is, it is a game to be played by teams of players. It is also possible to be played by individuals, but in that case the richness and benefits of the interaction with other players of your team are lost. It is important to emphasize that ProDec is a collaborative game, not a competitive one. This means that the group of players works collaboratively to win the game not to compete among them.

ProDec has also been developed to provide an automatic assessment of the performance of the players after a game play. This assessment is based on the assessment criteria set by the instructor.

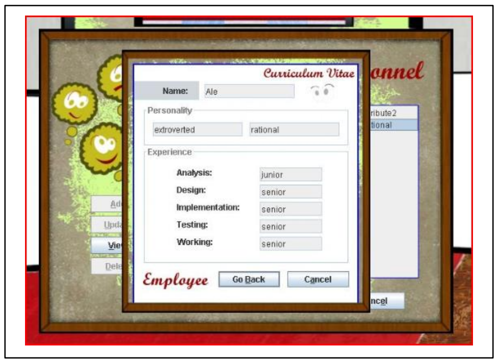
A. Objective

The aim of the game is to successfully manage a software project. The game is over when the project significantly overruns either the approved budget or the allocated time. During the game, the players have to plan a project, manage its execution and deal with the risks and unplanned events that may occur. They will succeed in the game if they are able to complete the project within the time and costs limits.

B. Basic Play

ProDec can be used in two different modes, namely, Full Play and Quick Play. When played in Full Play mode, the game allows the players to manage a software project they have previously planned. In this mode, the play is structured in three steps:

1. Onset. In this step, the player follows a process that guides them to make the project plan. The game helps the players to provide the information regarding the general data of the project, tasks definition, time and cost estimation, project team definition, personnel allocation to every task and risk estimation. It is important to highlight that for any member of the project team, the player has also to provide information about their professional experience and personality factors according to the sixteen personality factors described by Cattell [9], so that, during the play, it will be possible to simulate and assess how good or bad was the players’ decision during the team creation and task allocation. Figure 1 shows a screenshot of this process step focused on the personnel features.



Name: Ale

Personality: extroverted, rational

Experience:

Analysis: junior
Design: senior
Implementation: senior
Testing: senior
Working: senior

Figure 1. Screenshot of making the project plan

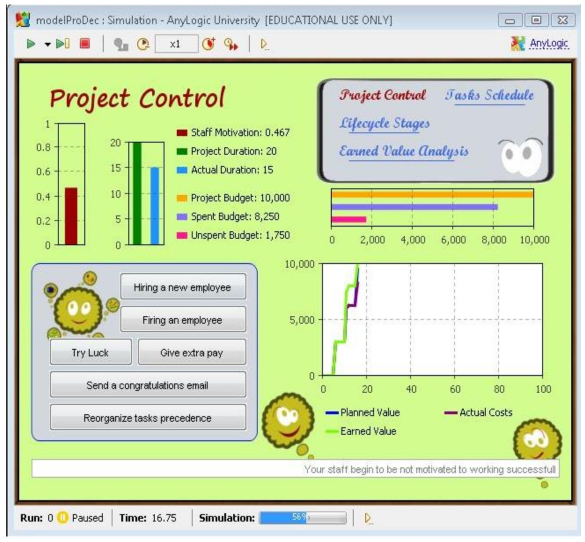
2. Execution. The second step consists on executing the project created in the first step. To do this, ProDec uses the information provided by the players to automatically generate the source code of a simulation model of the planned project. Once generated, the simulation model is run and the players start managing the project. The progress of the project depends on how well the project plan has been made, that is, the accuracy of the estimates of

time and cost, the quality and suitability of the project team, and the adequacy of the tasks allocated to the members of the project team. During the simulation of the project execution, the game shows the players a Control Screen where the progress of the project is shown as it can be seen in Figure 2. The following elements are shown in real time:

a. The time and budget spent and remaining.
b. The results of the earned value analysis of the progress of the project.
c. The level of the motivation of the project team.

Based on the progress of the project and their analysis of the situation, the players can make the following decisions:

a. Hire or fire a team member. In this case, every change in the project team will have a direct effect on the productivity because of the communication and training overheads derived from the team size, the contribution of the experience of the new or lost member, the overall team synergy, their motivation, etc.
b. Reorganize the project tasks. In this case, the players can reorganize the network of tasks which are yet to start. ProDec will check that the new network of activities is still consistent with the restrictions established to the tasks precedence in the project plan.
c. Send a thanks/congrats e-mail. According to the progress of the project, the player can decide to send a thanks note or congratulations e-mail to the project team members to, for instance, congratulate them for the consecution of a project phase on time and within budget. This will have a positive effect on the motivation of the team and, therefore, on their productivity. However, the game controls the unreasonable use of this option.
d. Give an extra payment. According to the available budget and the progress of the project, the player can also decide to give an extra payment to the project team. In this case, this action seeks to increase the external motivation of the team members leading, in some cases, to an increase in their productivity. Accordingly, this action will also reduce the available budget.
e. Try your luck. This option simulates the appearance of not planned risks. When selected, a random event takes place in the project. This event can have either a positive consequence, such as your sponsor increasing the budget, or a negative one, such as losing one of your team members because he decides to leave your company.



Your staff begin to be not motivated to working successfull

Figure 2. Project execute and control view

3. End. Once the simulation of the project execution is over, the last phase consists on the assessment of the players. By using the information that ProDec has been recording during the game play and the assessment criteria established by the instructor, ProDec generates an assessment report of the learners describing their level of achievement and uploads the results in the qualification book of Moodle, which is the Course Management System used in our subject currently. This report is mainly intended to be used by the instructor. In addition, ProDec prepares also this information to be provided in a very different format so that the learners get informed about their performance in a more engaging way. Basically, once a game play is over, ProDec automatically tweets a message to the Twitter account of the course telling about how the graces and disgraces of the team of future project managers. It also updates the Hall of Fame in the Facebook account of the course and gives a badge to those users who managed their project significantly well.

On the other hand, when the game is played in the Quick Play mode, the players go through a simplified onset phase, since the information describing the project plan has been provided by the instructor and they only need to select the project they want to manage among the ones already uploaded. The aim of this game mode is focused on the phase of execution of the project and the assessment of the management decisions. In this case, there is no need to assess the correctness of the project plan since it is assumed to be correct. Hence, the instructor, in this case, has also more options to establish the assessment criteria.

C. Lifecycle

Once the basics of the game have been described, it can be seen that the game helps the learner to see in action the group of processes of project management defined by the Project Management Institute (PMI) [10]. Figure 3 illustrates the relation between a game play's lifecycle and a project lifecycle. It can be seen that ProDec's Onset phase is related to the initiation and planning process groups, the Executing phase of the game with the cycle of executing, controlling and planning, and the End phase is related with the Closing process group proposed by PMI.

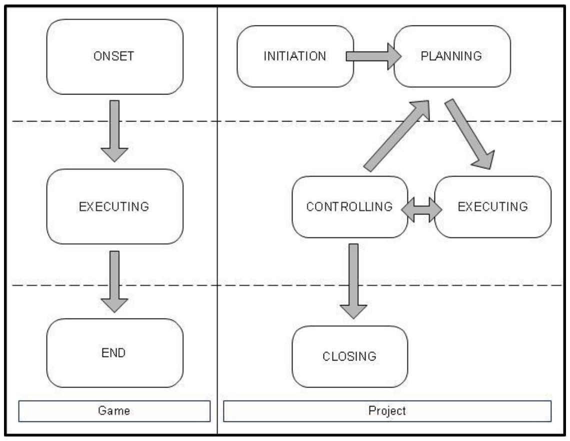


Figure 3. Lifecycle

D. Architecture

In order to address the functionality described above, ProDec has been developed using Java^{TM}, Anylogic^{TM} and MySQL^{TM} technologies. Figure 4 shows ProDec's architecture. As it can be seen, ProDec follows a three layer architecture. Two Java applications and the simulation model deal with the presentation and business layer, while two databases managed by MySQL^{TM} deal with the data layer.

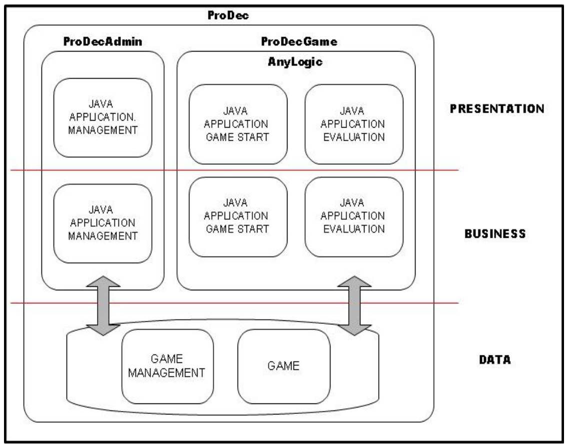

ProDec
ProDecAdmin
ProDecGame
AnyLogic
JAVA APPLICATION. MANAGEMENT
JAVA APPLICATION GAME START
JAVA APPLICATION EVALUATION
PRESENTATION
JAVA APPLICATION MANAGEMENT
JAVA APPLICATION GAME START
JAVA APPLICATION EVALUATION
BUSINESS
GAME MANAGEMENT
GAME
DATA

Figure 4. Architecture of ProDec

A description of the software applications follows:

a) ProDecAdmin is the software application that allows trainers to upload all the information required by the game. The trainers use this application to set the different game scenarios that can be played together with the rubrics for the players' assessment.
b) ProDecGame is the software application used by the players. This application is really composed of three applications:
    - An initial application that starts the game and dynamically generates the source file of the simulation model required to simulate the project.
    - A software application to simulate the execution of the project and allow for project monitoring.
    - A final application that finishes the game and performs the learners' evaluation by applying the rubric set by the trainer for the scenario that has been played.

IV. Learners' assessment

The process of assessing the learners' skills developed by playing the game is a process involving several elements belonging to different areas of ProDec. During the course of the game, the system saves records of the decisions made by the players during the simulation of the project, mainly as a response to a problem. In addition, ProDec also saves recurrently and autonomously records regarding the project status during the simulation and the initial estimates and risk analysis provided by the players at the beginning of the game. As a result, there are three sources of information for the application of the assessment criteria: a) the project plan with the initial estimates, b) the project monitoring data, and c) the kind and nature of the decisions the players made. Having these three sources of information about the learner performance, it allows the instructor to assess different types of skills.

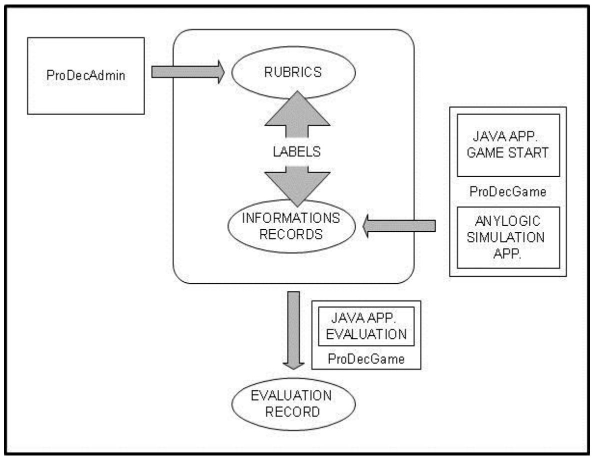


Figure 5. Elements of the assessment process

The assessment criteria are provided by the instructor in the form of a rubric by using ProDecAdmin. A rubric is structured in sections, each of which consists of an assessment criterion. An assessment criterion effectively links the information recorded in the rubric with the information recorded during the game by using a labeling system that matches the labels describing the skills that an assessment criterion with the records of the game that contain the information needed to assess such criterion.

As a consequence, ProDec is able to perform the learners' assessment by analyzing the information stored during the game and applying the assessment criteria set by the instructor, concluding with the generation of a detailed report, which describes the skills acquired by the players. This report allows learners and instructors to study the course of the played game, making it easier to analyze the decisions taken during the game and their results. Figure 5 shows the elements involved in the assessment process.

V. CONCLUSIONS AND FURTHER WORK

In this paper, we presented a serious game for software project management. The main objective is to take advantage of the engaging nature of games to place the learners in a virtual organization where they can manage software projects and solve real-life problems in a risk-free environment. The game accepts the information describing a software project plan and generates automatically a source code file with the equations of a simulation model to simulate the planned project. By running the simulation model, the game provides the players with the experience of seeing the effect of their planning and the decisions they are taking during the project execution. When the simulation of the project ends, the game performs an assessment of the learners according with the assessment criteria that the instructor has previously set.

The main contributions that make this game unique are the following:

1. The range of management options available for the instructor and learners to play with is larger than those of other similar tools. In fact, while other similar initiatives focus on practicing only certain aspects or techniques of project management, ProDec provides a training environment for learning the following:
    a. Project planning: Task identification, task time and cost estimates, among others.
    b. Project control and monitoring: Earned Value System and control scoreboards.
    c. Risk management: Quantitative risk analysis, incidence monitoring, unpredicted events and decision-making.
    d. Team management: Task allocation based on the experience of the team and the nature of the task, team motivation, team synergy, Brook's law [11], among others.
2. Dynamic and automatic generation of an ad-hoc simulation model. The information of the project plan is transformed into a set of equations of a discrete-event simulation model together with the source instructions that generate the user interface of the second phase of the game. Although using simulation at the core of a project management game is not an original feature, the available similar tools based on simulation models have a prebuilt simulation one. Hence, this kind of games provides only one scenario for simulation. ProDec, on the contrary, surpasses this limitation by creating an ad-hoc simulation model for every project plan the player can think of. Moreover, during the simulation, some of the decisions the player can make, can even change the equations of the simulation model in runtime.
3. There have been also other initiatives to apply serious games in software project management and in learning in general. However, most of these initiatives apply the benefits of game during the learning process only, forgetting the assessment part of every teaching and learning process. Very often, these experiences make use of the game to help learners learn, but they do not help instructors with the assessment, using the instructors more traditional assessment techniques for that phase. However, ProDec is intended to help also instructors with their assessment task by providing them with an environment where they can upload the assessment criteria for a project scenario that will be applied at the end of the game play to the data collected by ProDec during the play. The results of the assessment are also offered in several formats: as a report, as an update of the qualification book of Moodle and by different actions in the social networks used in the subject such as Twitter and Facebook.
4. The use of gamification elements. As a game, ProDec has been designed paying special attention to the user interface elements and the interactivity that can be expected in a game. In addition, some features coming from the gamification approach have also been added. These features, such as a Hall of Fame or a system of badges help to keep the learners engaged and motivated.

We can say that the game supports the three domains of Bloom's taxonomy: knowing, feeling and doing. Obviously, before playing the game the learners need to have studied the principles of the body of knowledge of software project management. This knowledge is put into practice by playing the game and having to evaluate the progress of the project and make decisions to achieve the initial objectives. Hence playing the game also helps to learn by doing. Finally, playing the game is also a social experience, since: a) the game is to be played in teams, and b) it also helps to share the results through social networks. These features, together with the engagement nature of games, transform the learning process into a social one where the feelings and emotions are naturally linked to the learning experience. We consider that the game also covers the six levels of Bloom's taxonomy. For example, at the lowest order process, ProDec helps the learners to remember what they have studied in their lectures about the software project management body of knowledge. To play the game, players also need to demonstrate they understand the facts they have studied, and they have to solve problems in new situations, such as estimating the budget of a new project, allocating tasks and making software teams with different treats of personality, or reacting to a risk they had never suffered before. The level of analysis is worked every time the player has to make a decision to improve the project results, since they need to carefully analyze the elements of the project, their relationships and the organizational principles that rule the progress of the project they are managing. After the analysis, players have to synthesize all the information into the decision they are going to make. Finally, the level of evaluation is achieved given the social nature of the game, where the players need to discuss, present their judgments and evidences that support the decision they would make, and then, negotiate with the rest of the members of their team about the decision to finally make.

Our aim is to build a tool for software project management learning as complete as possible. For this reason, our future works are aimed at two main objectives:

1. To perform evaluations of the current version of ProDec so that we can get the necessary feedback to design our following steps. We are currently working on this step with some evaluation sessions planned in different universities. So far, some evaluations have been made with one group of professors. During this academic course, we will conduct evaluation sessions with the students. In order to do this, we have based our evaluation process on the evaluation method developed at the Federal University of Santa Catarina [12].
2. To add new features to the game regarding software project management such as configuration management, change management, different methodologies of software development, among others.

ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Science and Technology with ERDF funds under grant TIN2010-20057-C03-03.

REFERENCES

[1] I. Ibrahim, “Teaching Project Management to IT Students: Methods and Approach,” 2nd International Conference on Education and Management Technology, IPEDR, Vol. 13, 2011, pp. 185 – 191. IACSIT Press, Singapore.

[2] C. Caulfield, J.C. Xia, D. Veal, and S.P. Maj, “A systematic survey of games used for software engineering education,” Modern Applied Science, Vol. 5, No. 6, Dec. 2011, pp. 28-43.

[3] C. Caulfield, D. Veal, and S.P. Maj, “Teaching software engineering project management-A novel approach for software engineering programs,” Modern Applied Science, Vol. 5, No. 5, Oct. 2011, pp. 87-104.

[4] E.O. Navarro and A. Van Der Hoek, “SimSE: An interactive simulation game for software engineering education,” Proceedings of the Seventh IASTED International Conference on Computers and Advanced Technology in Education, 2004, pp. 12-17.

[5] C.G. Von Wangenheim, R. Savi, and A.F. Borgatto, “DELIVER! – An educational game for teaching earned value management in computing courses,” Information and Software Tecnhnology, Vol. 54, No. 3, Nov. 2012, pp. 286-298.

[6] A. Drappa and J. Ludewig, "Simulation in software engineering training," Proceedings of the 22nd international conference on Software engineering, ACM, 2000, pp. 199-208.

[7] L.W. Anderson and D.R. Krathwohl (Eds.), “A Taxonomy for Learning, Teaching, and Assessing: a Revision of Bloom's Taxonomy of Educational Objectives,” Longman, New York, 2001.

[8] D.R. Krathwohl, “A revision of Bloom's taxonomy: An overview,” Theory into Practice, 41 (4), 2002, pp. 212-218.

[9] R.B. Cattell, H.W. Eber, and M.M. Tatsuoka, “Handbook for the sixteen personality factor questionnaire (16 PF),” 1988. http://www.getcited.org/pub/102817845 [rerieved: August, 2013].

[10] Project Management Institute, “A Guide to the Project Management Body of Knowledge (PMBOK® Guide)”, Fifth Edition, Project Management Institute, 2013.

[11] F.P. Brooks, “The Mythical Man-Month,” Addison-Wesley Longman Publishing Co. Inc., Boston, MA, USA, anniversary ed. edition, 1995.

[12] R. Savi, C.G. Von Wangenheim, and A.F. Borgatto, “A Model for the Evaluation of Educational Games for teaching Software Engineering,” 25th Brazilian Symposium on Software Engineering, 2011, pp. 194-203.

[13] XJ Technologies. Anylogic^{TM}. http://www.anylogic.com/ [retrieved: August, 2013].

Open Source Legality Compliance of Software Architecture

A Licensing Profile Approach

Alexander Lokhman, Antti Luoto, Imed Hammouda, and Tommi Mikkonen
Tampere University of Technology, Department of Pervasive Computing
Tampere, Finland
firstname.lastname@tut.fi

Abstract — **The architecture of a software system is typically described from multiple viewpoints, such as logical, process, and development views. With the increasing use of open source components, there is a new emerging view that should be taken into account: the legality view. The legality view makes explicit the legality concerns of software architecture such as Intellectual Property Rights (IPR) issues and use/distribution terms of the components. These issues are particularly important, when they impose architecturally significant requirements that may influence the architecture. In this paper, we discuss the compliance of software architecture with respect to the legality aspects of open source licenses, and address the various facets of open source legality compliance. We then propose a Unified Modeling Language (UML) profile-based approach and tool to address the legality concerns of open source at the level of software architecture. The technique has been applied to express and analyze the legality view of an industrial case study.**

Keywords-UML profiles; open source software; licensing; software architecture

I. INTRODUCTION

Software architecture has been standardized as the fundamental organization of a system embodied in its components, their relationships to each other and to the environment, and the principles guiding its design and evolution [26]. Commonly identified stakeholders of software architecture include testers, product managers, users, designers, marketing personnel, and so forth. Architecturally significant requirements resulting from these perspectives commonly include quality attributes such as testability, scalability, understandability, modularity, flexibility, and so on. Thus, the architecture of a software system is represented by multiple views [11]. These views vary in nature and are complementary to each other. Some views show the organization of the code units (e.g., packages and classes). Others show the runtime view of the system (e.g., processes and threads). A third view is to explain how the system is deployed on physical hardware (the deployment view). Each architecture view defines the types of elements and relations that can be represented in that view, and provides means for reasoning about their properties.

We claim that there is a new emerging view to any software system that should be taken into account increasingly often: the legality view. The goal of the legality view is to make explicit the legality concerns of software architecture – such as Intellectual Property Rights (IPR) issues and use/distribution terms of the individual components – in particular when legal aspects are architecturally significant and should therefore influence the architecture. In this spirit, the legality view should clearly state how the legality constraints of the individual architectural elements are satisfied by the overall architecture.

So far, the legality view has been considered in designs to some extent, for instance in terms of encryption and safety requirements (as part of the non-functional view) or data privacy issues (as part of the data view), just to list a few examples. However, due to the increasing use of Free/Libre/Open Source Software (FLOSS) systems freely available on the Internet (e.g., [20]), where licensing issues differ from the conventional proprietary setting and concern the very core of software design, a more holistic view of legality issues associated with open source components is needed [1, 10, 23].

Within the wide spectrum of legality issues of software architecture, the main focus of this paper is to examine open source licenses as primary source for legality concerns in software solutions that involve both proprietary and open source components. We argue that the terms dictated by open source licenses may constrain the architecture of a software system and even act as an architectural driver during design. For software architects, being aware of the rights and duties of the licenses is crucial in producing an acceptable system from the legality perspective. This is an important yet often overlooked piece of the architecture puzzle, which explicitly communicates the architecture's legality fitness for the purpose of providing all the stakeholders with the confidence that the software system does not suffer from licensing violations and shortcomings.

The contribution of the paper is threefold: First, we review the main factors that shall be taken into consideration when addressing the legality compliance issue of FLOSS intensive systems. Second, we introduce the concept of a licensing profile, which is a Unified Modeling Language (UML) profile [13] used to capture the licensing rules and constraints dictated by FLOSS licenses and expressed in architectural design expressed in UML. Third, we present a generic tool named Open Source Software Licensing (OSSLI) [21] that allows for working with licensing profiles.

The rest of the paper is structured as follows. In Section 2, we give a discussion on the legality tensions that arise in

FLOSS intensive systems and discuss the significance of representing legality concerns in architectural designs. In Section 3, we discuss in detail the concept of licensing profiles. A concrete tool environment for licensing profiles is then presented in Section 4. In Section 5, we introduce a real-life design, where the legality view has been incorporated in development from the very beginning to demonstrate the feasibility of the approach. In Section 6, we discuss our approach related to existing works. Finally, in Section 7, we conclude and point out directions for future work.

II. Managing Open Source License In Architectural Design Models

In this section, we review licensing constraints dictated by open source licenses and their significance to architectural design.

A. *Legality Tension of FLOSS Intensive Systems*

When addressing the legality compliance issue of FLOSS intensive systems, there are a number of factors that must be taken into account. These factors not only stem from the nature and terms of the licenses themselves, but also are related to the way the subject software is implemented, packaged, and deployed.

There are plenty of licenses and license models. A straightforward observation when working with open source licenses is that there are many of them – the Open Source Initiative (OSI) [18] lists about 70 licenses. Popular licenses include the GNU General Public License (GPL), the Lesser GNU General Public License (LGPL), the Apache license, the Massachusetts Institute of Technology license (MIT), and the Berkeley Software Distribution license (BSD). The terms of different licenses vary considerably. To give an example, some licenses such as MIT are classified as permissive, granting very broad rights to licensees and allowing almost unlimited use of the licensed code. Other licenses such as GPL are classified as strong copyleft, requiring that works based on the licensed code be published and relicensed to others on the same terms of the initial license. In the middle are weak copyleft licenses such as LGPL, which is a compromise between permissive licenses and strong copyleft. The LGPL grants flexibility to users when linking to licensed software libraries. However, any modifications to the original library should be contributed back on the same terms of the license. Moreover, some licenses have several versions, and there are subtle changes between different versions. A good example is the case of GPL v2 and GPL v3, which are not fully compatible with each other. In addition, the list is by no means complete, and new licenses can be introduced if so desired. For example, a new license can add some minor differences to an earlier one, thus generating a discrepancy between the licenses, or a completely new license can be introduced.

Licenses can be conflicting [5, 8]. To give an example of possible legal incompatibilities between software components, Table I presents a number of open source licenses and their compatibility properties (across open source components themselves) categorized into three cases: mixing and linking is permissible, only dynamic linking is permissible, and completely incompatible.

As an example, a software component under the terms of GPL cannot be directly linked with another under the terms of the Apache license. In this case, the main reason is that GPL'ed software cannot be mixed with software that is licensed under the terms of a license that imposes stronger or additional terms, in this case the Apache license. The Apache 2.0 license allows users to modify the source code without sharing modifications, but they must sign a compatibility pledge promising not to break interoperability, which fundamentally contradicts GPL terms.

TABLE I. Example Open Source Licenses and their Compatibility

	PHP	Apache	IPL	SSPL	Artistic
GPL	3	3	3	1	3
LGPL	2	2	2	1	2
BSD	1	1	1	1	1

1- Mixing and linking permissible
2- Only dynamic linking is permissible
3- Completely incompatible

Is it derived or combined work? When integrating third party open source components, possibly together with own work, the restrictions and obligations, which the used licenses impose, may depend on whether the work is considered as derived (derivative) or combined (collective) [6]. A simple example of derived work is a modified version of the original software. However, the distinction between derived and combined works becomes trickier when producing new work by combining or linking multiple software components, possibly distributed under the terms of different licenses. Take the example of a software system S, which is the result of linking together an open source component C1 and an own developed component C2. A common interpretation is that system S is considered to be derived work if C1 and C2 link statically (linked during compile or build time) and that S is considered to be combined work if C1 and C2 link dynamically (the two libraries are loaded into a client program at runtime). In a typical case, however, the judge in a court of law makes the final decision. As a matter of fact, the court decision might depend on the specific legal framework of the jurisdiction, in which the case arises, resulting in even more complex legality issues for software developers.

There are thousands of open source components with different risk levels depending on their usage scenario. The number of open source components has grown at an exponential rate during the last decade. This has given software developers a jump on creating software based on existing code. However, many companies are reluctant to use open source software due to the legal risks associated with the use of those components. There have been attempts to classify open source components according to their risk level [7, 28]. Table II gives an example categorization. Four usage scenarios are identified: using the component as a redistributable product, as part of service offering, as a development tool, and for internal use. Three levels of risks have been proposed, as described in the following.

According to von Willebrand and Partanen, [28], valid means that the package can be used as instructed and that no risk has been identified. Possible risk means an interpretation question has been found. This type of issues can be solved by either 1) removing/replacing the problematic files or 2) acquiring additional permissions from the respective right holder or 3) not using the package at all or 4) based on the particular company's risk preferences in such project, a company could accept the risk. Legally, an interpretation question means that an eventual realizing risk would be civil law risk, e.g., monetary (not criminal). Clear risk means that a risk that cannot be interpreted in a way that would not include the risk has been found. This type of issues can be solved only by 1) removing/replacing the problematic files or 2) acquiring additional permissions from the respective right holder or 3) not using the package at all. A company normally cannot accept this type of risk, since it means the possibility of not only civil law risks, but criminal risks. As an example, component Agent++ can be used internally with no risk, has a possible risk when used as a development tool, but exhibits a clear risk when used as part of service offering or a redistributable product.

TABLE II. EXAMPLE SOFTWARE COMPONENTS AND THEIR RISK LEVELS

Comp.	License	Redistribution	Service offering	Development tool	Internal use
Agent++	Agent++ license	3	3	2	1
SwingX	LGPL	3	3	3	3
Libxml2	MIT	1	1	1	1
Cglib	Apache	2	1	1	1

(1) Valid (2) Possible risk (3) Clear risk

Open Source legality interpretations are subject to the way software is implemented, packaged, and deployed [8, 16]. The legality requirements imposed by FLOSS licenses, such as the requirement to publish source code (i.e. the copyleft rule of GPL), may depend for instance on the interaction type of the components (data-driven versus control-driven communication). In the case of mere data exchange between components, there is no copyleft obligation as the two components are considered as separate programs. Also, the copyleft obligation of GPL does not hold if the FLOSS component (or a modified version of it) is deployed as a hosted service. However, if the hosted code is licensed under the terms of AGPL (Affero General Public License) [29], the copyleft requirement does hold, but only in the case of user interaction with the hosted service (in contrast to service to service interaction). In addition, the copyleft requirement of GPL may not hold in case of interactions through standardized interfaces such as the use of operating system public Application Programming Interface (API), in contrast to system hacks that make the two communication components strongly coupled. Finally, compatibility concerns among different licenses may be circumvented if the packaging of components is done by the user instead of building the entire system at the vendor site.

B. Significance of Legality Concerns in Architectural Design

This work advocates for the usefulness of representing open source legality concerns in architectural design. This would allow addressing the licensing issues early in the development process. Accordingly, we foresee the following benefits of the approach:

- Raising the awareness of licensing issues for software architects. This could be achieved by offering a communication medium for software architects with respect to legality matters.
- Using architectural models as an early simulation medium with respect to license integrity and validity, which allows the possibility to detect possible violations.
- Aligning and keeping source code and architectural design in sync from the viewpoint of software licenses. This prevents architectural erosion with respect to licensing decisions.
- Legality constraints can be exploited in a forward engineering scenario, for instance to suggest possible architectural solutions to overcome detected license violations. In addition, the constraints can be used to provide guidelines for component selection with respect to possible licenses that can be used.
- Allowing the ability to organize architectural design into license independent models and license specific models to better analyze the effect of licensing decisions.
- Providing a better way of visualizing license violations and their context. It is beneficial to view the violations in graphical models rather than textual source code.
- Studying how the terms of software licenses can influence quality attributes like scalability (e.g., number of users), which are often considered at the architectural level.

According to these points, we propose a visual modeling based approach that enables analyzing license related problems in early development phases while reusing existing models. The approach is designed to work with and support architectural design made in UML.

III. A PROFILE BASED APPROACH

In this section, we present our approach for documenting the legality view of software architecture, assuming that the design model is expressed in UML. Accordingly, we introduce the concept of licensing profiles in detail and illustrate the concept with two example profiles. We start with a brief introduction to UML profiles.

A. UML Profiles

The generality of UML constrains its applicability for modeling narrow-scaled domains or problem fields. However, UML offers mechanisms for extending the language. With the help of these mechanisms, it is possible to create an extension that adds more expression power to UML on a certain field or environment. In addition, traditional UML can be hidden on a lower level so that only relevant properties are displayed.

One of the extension mechanisms in UML is the light-weight profile mechanism, which is based on meta-modeling [13]. Profiles are packages that contain stereotypes, tagged values, and constraints. Stereotypes are a special kind of meta-classes while tagged values are meta-attributes of those classes. Meta-class is a type defined by UML specification. Based on these features, it is possible to define a Domain Specific Modeling Language (DSML) for a certain application field. Profiling is a mechanism of UML, and thus the definitions do not necessarily reflect the actual implementation of the problem but provide a way to express issues conveniently. For example, it is difficult to say how a stereotyped class is implemented in real life, but as a modeling tool it is a convenient way to visualize information.

B. Licensing Profiles

A licensing profile is a UML profile used to attach IPR related information to UML models. Licensing profiles introduce concepts related to the properties of open source licenses in the form of stereotypes and meta-attributes. This allows the user to create a UML model that takes into account what licenses each component is associated with. For example, a software package could be annotated with information such as copyright holder, license type, and the risks associated with its use in different usage scenarios.

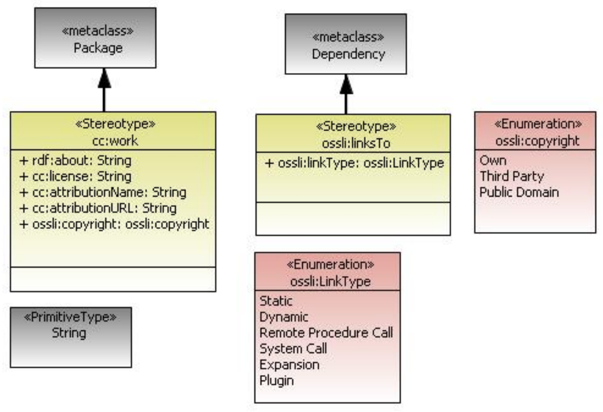


Figure 1. CC REL profile

Figure 1 depicts a licensing profile that is partly based on the specification of the Creative Commons Rights Expression Language (CC REL), a semantic ontology for modeling licenses [2]. The profile makes use of Resource Description Framework (RDF) descriptions for modeling licenses. In addition, the profile introduces concepts, attributes and stereotypes not found in the original CC REL. This is reflected in the naming strategy of the profile – ”cc:” refers to CC REL concepts whereas “ossli” refers to other concepts developed in this work.

For example, the profile defines a stereotype named *cc:work* that corresponds to CC REL class *Work*. The class is defined as “a potentially copyrightable work” in CC REL description. Table III shows the tagged values of *cc:work*. As an example of concepts outside CC REL, the profile defines one stereotype for dependencies. The stereotype is named *ossli:linksTo* and contains one tagged value called *ossli:LinkType*. The tagged value's range is defined in enumeration *ossli:LinkType*. With this tagged value, it is possible to choose a linking type from multiple common types such as static, dynamic, remote procedure call, etc. With the help of CC REL profile, it is possible for example to tell why two open source licenses are conflicting by examining the RDF definition of the license.

TABLE III. TAGGED VALUES OF CC:WORK

Tagged value	Type	Description
rdf:about	String	A standard way in RDF for defining the resource being described. (Uniform Resource Identifier) URI.
cc:license	String	URI to RDF definition of the license.
cc:attributionName	String	The name the creator of a Work would prefer when attributing re-use.
cc:attributionURL	String	The Uniform Resource Locator (URL) the creator of a Work would prefer when attributing re-use.
ossli:copyright	ossli:copyright	Copyright status of the package defined by enumeration ossli:copyright.

A more advanced licensing profile, named OSSLI profile, is depicted in Figure 2. The profile is based on the specification of Software Package Data Exchange (SPDX) [25], recommendations by OSI and other de facto rules for package compliance review [28].

TABLE IV. TAGGED VALUES OF LICENSEDPACKAGE

Tagged value	Type	Description
Copyright	String	Copyright information in free text format.
Description	String	Description of the package in free text format.
License	LicenseType	One or more licenses.
Redistribution	Validity	Validity for redistributing the package.
Development Tool	Validity	Validity for using the package as a development tool.
Service	Validity	Validity for offering functionality as a service.
Internal Use	Validity	Validity for using the package internally.
ID	Integer	Identification for the package.
Ownership	OwnershipType	Ownership of the package.

A fundamental concept in the profile is the stereotype *LicensedPackage*, which extends the standard UML package. *LicensedPackage* has multiple tagged values that are introduced in Table IV. The Tagged values with the type *Validity* are based on package compliance review [28]. Enumeration *Validity* is defined using four values: *Valid*, *Possible Risk*, *Clear Risk* and *Unknown*. The supported licenses are listed in *LicenseType* enumeration, which includes *Unknown* for packages with unknown license or unexpressed license information. *OwnershipType* is defined in the profile as an enumeration with three values: *Own*, *ThirdParty*, *PublicDomain* and *Unknown*.

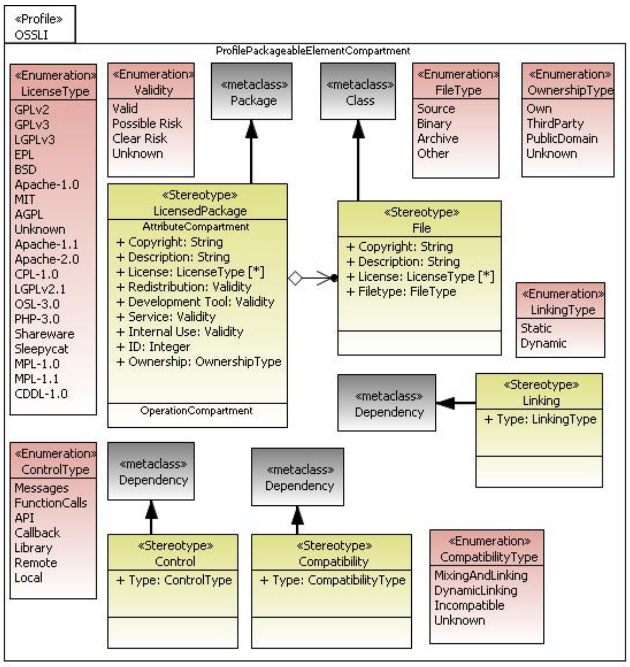


Figure 2. OSSLI profile

The profile shows that *LicensedPackage* is composed of classes that are stereotyped as *File*, which have own tagged values. In addition, the profile defines three dependency stereotypes. *Linking* stereotype consists of one tagged value named *Type*, which tells whether the linking between packages is static or dynamic. Thus, *Type* is defined by enumeration *LinkingType* with values *Static* or *Dynamic*. *Control* stereotype describes control type between packages, such as if the packages communicate with each other using API or remote procedure calls. *Compatibility* is a stereotype designed to mark the compatibility mode of licenses as described previously in Table I.

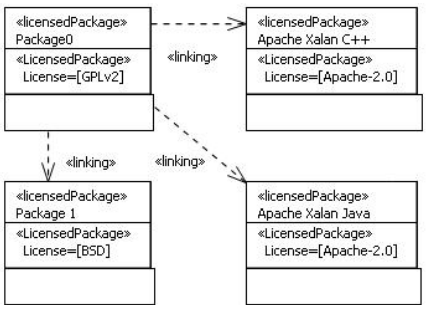

«licensedPackage» Package0
«LicensedPackage» License=[GPLv2]
«licensedPackage» Apache Xalan C++
«LicensedPackage» License=[Apache-2.0]
«licensedPackage» Package 1
«LicensedPackage» License=[BSD]
«licensedPackage» Apache Xalan Java
«LicensedPackage» License=[Apache-2.0]
«linking»
«linking»
«linking»

Figure 3. Illustrative example model using OSSLI profile

An illustrative example model using the OSSLI licensing profile is shown in Figure 3. The example consists of four software packages, of which two are owned packages (*Package0*, *Package1*) and two are third party packages (*Apache Xalan C++*, *Apache Xalan Java*). Package0 is linked to all the other packages. The profiled model exhibits licensing information such as license used and linking type information between packages.

IV. OSSLI TOOL ENVIRONMENT

In order to illustrate the use of licensing profiles, a tool named OSSLI [21] has been developed on top of Papyrus modeling environment [22]. The tool is capable of documenting licensing information and managing open source legality concerns in architectural design. In OSSLI, design models are expressed as profiled UML package diagram. Figure 4 depicts the user interface of OSSLI showing the example design model introduced in Figure 3 (middle part of the figure). The left part of Figure 4 shows the selection of the OSSLI licensing profile selected for application.

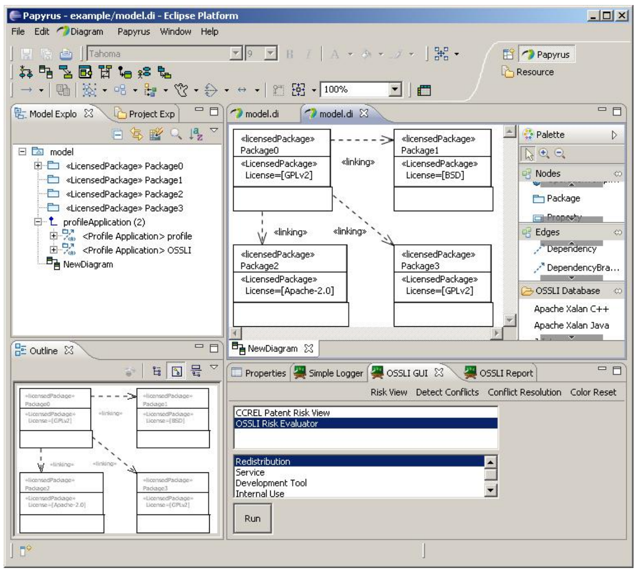

Papyrus - example/model.di - Eclipse Platform
File Edit Diagram Papyrus Window Help
Papyrus
Resource
Model Explo
Project Exp
model.di
model.di
model
«LicensedPackage» Package0
«LicensedPackage» Package1
«LicensedPackage» Package2
«LicensedPackage» Package3
profileApplication (2)
<Profile Application> profile
<Profile Application> OSSLI
NewDiagram
«licensedPackage» Package0
«LicensedPackage» License=[GPLv2]
«licensedPackage» Package1
«LicensedPackage» License=[BSD]
«licensedPackage» Package2
«LicensedPackage» License=[Apache-2.0]
«licensedPackage» Package3
«LicensedPackage» License=[GPLv2]
«linking»
«linking»
«linking»
Palette
Nodes
Package
Edges
Dependency
DependencyBra...
OSSLI Database
Apache Xalan C++
Apache Xalan Java
Outline
NewDiagram
Properties
Simple Logger
OSSLI GUI
OSSLI Report
Risk View
Detect Conflicts
Conflict Resolution
Color Reset
CCREL Patent Risk View
OSSLI Risk Evaluator
Redistribution
Service
Development Tool
Internal Use
Run

Figure 4. OSSLI user interface

In addition, the bottom part of Figure 4 shows a scenario of running a risk evaluator for product redistribution on the example model. Figure 5 shows the results of the risk evaluation. *Package0* has been reported as risky (marked with red color) while all other packages are without risks (marked with green color). Alternatively, the user could run risk evaluation with respect to service offering, development tool or internal use. The analysis is based on the information of *LicensedPackage*'s tagged values included in the OSSLI profile and introduced in Table II.

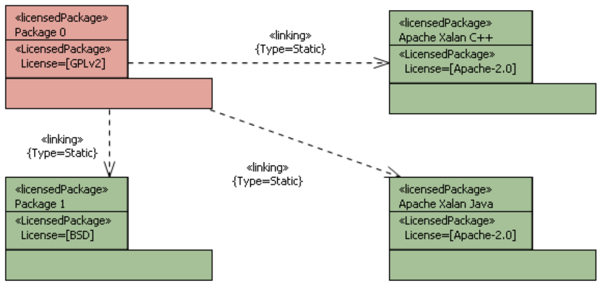

«licensedPackage» Package 0
«LicensedPackage» License=[GPLv2]
«linking» {Type=Static}
«licensedPackage» Apache Xalan C++
«LicensedPackage» License=[Apache-2.0]
«licensedPackage» Package 1
«LicensedPackage» License=[BSD]
«licensedPackage» Apache Xalan Java
«LicensedPackage» License=[Apache-2.0]

Figure 5. Example risk evaluation of packages

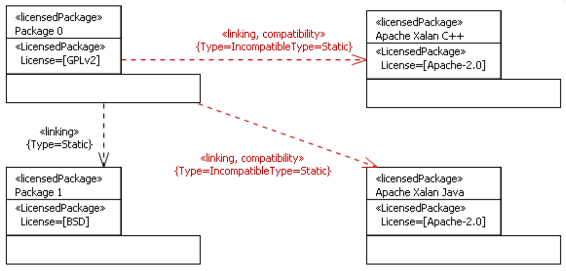

«licensedPackage» Package 0
«LicensedPackage» License=[GPLv2]
«linking, compatibility» {Type=IncompatibleType=Static}
«licensedPackage» Apache Xalan C++
«LicensedPackage» License=[Apache-2.0]
«linking» {Type=Static}
«licensedPackage» Package 1
«LicensedPackage» License=[BSD]
«licensedPackage» Apache Xalan Java
«LicensedPackage» License=[Apache-2.0]

Figure 6. Example of conflict detection

The user could also perform license conflict detection on the profile model. A license conflict occurs when connecting components to each other. This is illustrated in Figure 6. Detected conflicts are presented to the user and are highlighted in the UML diagram in red color. In the figure, *Package0* is reported as conflicting with packages *Apache Xalan C++* and *Apache Xalan Java*. The conflict is reported based on the compatibility values shown in Table 1 and represented using the *Compatibility* stereotype in the OSSLI profile.

V. CASE STUDY: SOLA

The proposed legality view to software architecture has been incorporated in the development of a real-life open source system known as Solutions for Open Land Administration (SOLA) [3]. The project, which is supported by Food and Agriculture Organization (FAO), aims to implement an open source land registration and administration system that will be deployed in at least three developing pilot countries – Nepal, Ghana, and Samoa. The role of the authors of this paper is to provide open source consulting support related to the development of the system, software review, and community building.

The development of the system has been organized in two main phases, a generic phase where the core components of the system are developed by a closed team, and an application phase where the system is adapted to the contexts of the three countries and released to the open source community for further development. A basic project requirement was to reuse the maximum number of existing open source components. This has led to the adoption of tens of open source components with different open source licenses. In addition, a number of other components have been developed by the project team. Figure 7 depicts a fragment of the SOLA system architecture.

As part of the software review task, we have assessed the system architecture from a legality perspective. Example questions we had to address include:

1. Could a GPL'ed icons library be used in the presentation layer?
2. How should the components developed by FAO be licensed? Both individually and as the whole SOLA package?
3. Are there any compliance violations among component interactions?
4. Could the SOLA components be used in proprietary products? If not, how to circumvent this issue?
5. Are there any legality problems related to software compliance with the national e-gov strategies of the pilot countries?

As example answers to the above questions, it was deemed risky to use a GPL'ed icons library as this would trigger the copyleft obligations of GPL, which would be a problem in case the software is used in proprietary systems. Therefore, the library has been discarded.

Figure 8 shows a compliance exercise session for SOLA design model in the OSSLI tool. Analyzing the components interactions and their licenses, several important findings have been observed. First, we identified all possible legality incompatibilities. In Figure 7, a possible risk is mixing LGPL'ed *JasperReports* library and Apache Licensed *Barcode4J*. According to the terms of the licenses developers should use dynamic linking in order to achieve more independence among these components. Other conflict detection risks are highlighted in Figure 8.

As for the components written by the FAO team, we proposed the use of the modified BSD license because it is compatible with all other internally used licenses. Another option we have discussed is to use to use GPL v2. However, the latter option would bring clear risks when combining GPL'ed packages with Apache Licensed libraries (e.g., *Dozer*, *MyBatis*). This is because Apache License and GPL are completely incompatible.

Finally BSD license was also proposed as the main license of the entire SOLA system. This minimizes the legality risks when adopting the software in the pilot countries, and allows commercial companies to develop proprietary software on top of the SOLA system and its components. Furthermore, no conflicts were found between the proposed license scheme and the guidelines of the national strategies of the pilot countries.

VI. RELATED WORK

The fashion FLOSS components are allowed to interact with each other and proprietary software has become an important architectural concern. Present design approaches optimized for the technical aspects of software architecting, such as scalability, reusability, and testability and tend to diminish or even completely overlook the legality dimension

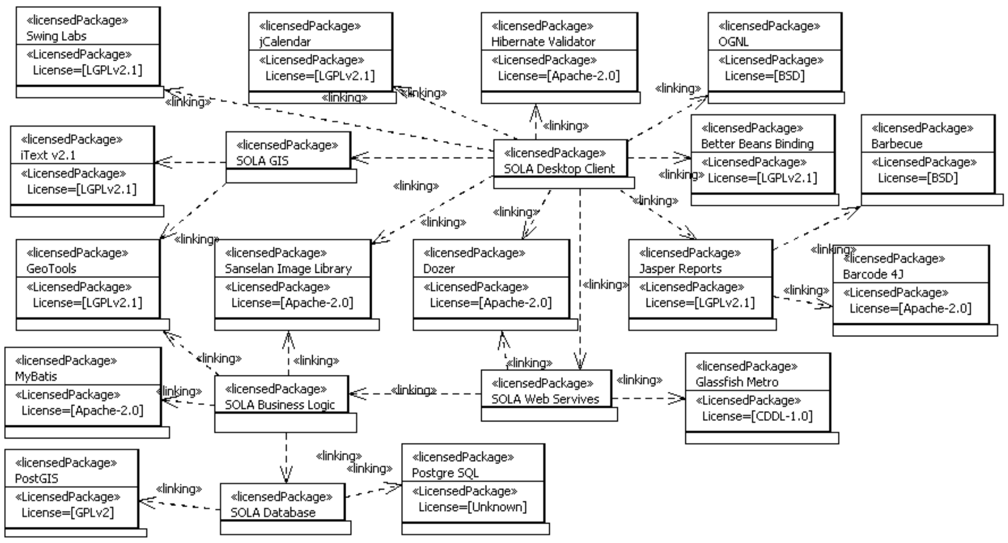
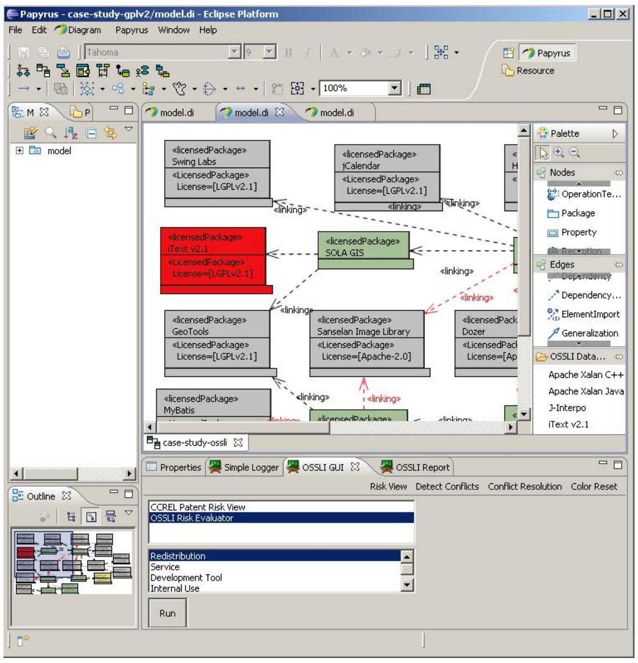

«licensedPackage» Swing Labs «LicensedPackage» License=[LGPLv2.1]
«licensedPackage» jCalendar «LicensedPackage» License=[LGPLv2.1]
«licensedPackage» Hibernate Validator «LicensedPackage» License=[Apache-2.0]
«licensedPackage» OGNL «LicensedPackage» License=[BSD]
«licensedPackage» iText v2.1 «LicensedPackage» License=[LGPLv2.1]
«licensedPackage» SOLA GIS
«licensedPackage» SOLA Desktop Client
«licensedPackage» Better Beans Binding «LicensedPackage» License=[LGPLv2.1]
«licensedPackage» Barbecue «LicensedPackage» License=[BSD]
«licensedPackage» GeoTools «LicensedPackage» License=[LGPLv2.1]
«licensedPackage» Sanselan Image Library «LicensedPackage» License=[Apache-2.0]
«licensedPackage» Dozer «LicensedPackage» License=[Apache-2.0]
«licensedPackage» Jasper Reports «LicensedPackage» License=[LGPLv2.1]
«licensedPackage» Barcode 4J «LicensedPackage» License=[Apache-2.0]
«licensedPackage» MyBatis «LicensedPackage» License=[Apache-2.0]
«licensedPackage» SOLA Business Logic
«licensedPackage» SOLA Web Services
«licensedPackage» Glassfish Metro «LicensedPackage» License=[CDDL-1.0]
«licensedPackage» PostGIS «LicensedPackage» License=[GPLv2]
«licensedPackage» SOLA Database
«licensedPackage» Postgre SQL «LicensedPackage» License=[Unknown]
«linking»

Figure 7. The SOLA Project Legality View



CCREL Patent Risk View
OSSLI Risk Evaluator

Redistribution
Service
Development Tool
Internal Use

Figure 8. Risk evaluation and conflict detection in SOLA model

that is becoming increasingly important to manage legal dependencies of open source components.

The legality challenge of FLOSS has been partly addressed using so-called license analysis techniques and tools (Table V). Some of the tools provide functionality to identify the licenses through source code analysis. Examples of these tools are Fossology [4], Automated Software License Analysis (ASLA) [27], and Ninka [17]. LChecker [12] provides a similar functionality but takes a slightly different approach. It utilizes Google Code Search service to check if a local file exists in a FLOSS project and if the licenses are compatible. In addition to license identification, Open Source License Checker (OSLC) [19] also provides support for license conflict detection in source code. Dependency Checker Tool (DCT) [14] focuses on detecting compliance problems at static and dynamic linking level on binaries, based on predefined linking and license policies.

TABLE V. A COMPARISON OF OPEN SOURCE LICENSE MANAGEMENT TOOLS

.	Source analysis	License identification	Design analysis	Conflict detection
Ninka	Yes	Yes	No	No
ASLA	Yes	Yes	No	No
Fossology	Yes	Yes	No	No
LChecker	Yes	Yes	No	No
OSLC	Yes	Yes	No	Yes
DCT	No	No	No	Yes
Qualipso	No	No	OWL	Yes
ArchStudio4	No	No	Custom	Yes
OSSLI	No	No	UML	Yes

Compared to the OSSLI tool, the above technique are mostly useful in analyzing ready packaged software systems but give little guidance, with respect to licensing issues, for software developers during the development activity itself. A number of other tools, such as [23] and [1] do provide support for analyzing license conflicts at the architectural level. However, these tools generate own architectural views and have limited integration with the artifacts that software architects work with. The former uses Web Ontology Language (OWL) for modeling open source licenses and the latter uses a custom formal approach. Furthermore, these tools fall short in their ability to support a number of important practices related to license compliance checking. For example, decisions made during the process of fixing the legality compliance problems in the software architecture could also be recorded for future recommendations [15].

There are a number of ontologies and standards proposed for documenting the legal rules and constraints of software systems. Examples include Legal Knowledge Interchange Format (LKIF) [9], Software Package Data Exchange (SPDX) [25], and QualiPSo Intellectual Property Rights Tracking (IPRT) [23]. These works could contribute to the foundation of the proposed legality view, but nevertheless should be enhanced for better ties with the work processes and methods of software architects.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we have proposed a new perspective to the software architecture, the legality view. The goal of the view is to make explicit the legality concerns of software architecture such as IPR issues and use/distribution terms of the components, which are often important concerns in all software, but need to be further emphasized in open source development due to the different licensing schemes. The view is particularly important in cases where the legality view introduces architecturally significant requirements. In the paper, the benefits of the view were first demonstrated by a small illustrative example and a real-life design, where the different FLOSS related concerns play an important role in the design of architecture.

The consequences of the introduction of a new view to software architecture are many. To begin with, the complexity of legal issues and their effect in software design becomes visible. While making such issues explicit on one hand helps designers to take them into account, on the other hand the design methods and practices must be revised to precisely reflect the new view in an integrated fashion.

In order to express the discussed legality view in a practical fashion, we have proposed the concept of licensing profile, an adaptation of the UML profile concept for the modeling of open source licensing rules and constraints. We then presented tool support for working with licensing profiles.

As future work, we plan to use licensing profiles as a basis for building novel techniques to devise optimal architectural solutions taking into consideration the legality constraints. This could be achieved, for instance, through the use of genetic algorithms [24].

REFERENCES

[1] T. A. Alspaugh, H. U. Asuncion, and W. Scacchi, “Analyzing Software Licenses in Open Architecture Software Systems,” Proc. FLOSS 2009, 2009, pp. 54–57.

[2] Creative Commons. Describing Copyright in RDF. http://creativecommons.org/ns. Last accessed June 2013.

[3] FAO. Solutions for Open Land Administration. http://flossola.org/. Last accessed June 2013.

[4] FOSSology. http://fossology.org/. Last accessed June 2013.

[5] D. M. German, M. Di Penta, and J. Davies, “Understanding and Auditing the Licensing of Open Source Software Distributions,” Proc. ICPC 2010, 2010, pp. 84–93.

[6] D. M. German and A. E. Hassan, “License Integration Patterns: Addressing License Mismatches in Component-based Development, “ Proc. ICSE 2009, May. 2009, pp. 188–198.

[7] F. P. Gomez and K. S. Quiñones, “Legal Issues Concerning Composite Software, “ Proc. ICCBSS 2008, 2008, pp. 204–214.

[8] I. Hammouda, T. Mikkonen, V. Oksanen, and A. Jaaksi, “Open Source Legality Patterns: Architectural Design Decisions Motivated by Legal Concerns”, Proc. AMT 2010, Tampere, Finland, ACM Press, October. 2010, pp. 207–214.

[9] R. Hoekstra, J. Breuker, M. Di Bello, and A. Boer, “The LKIF Core Ontology of Basic Legal Concepts, “ Proc. LOAIT 2007, 2007, pp. 43–63.

[10] International Free and Open Source Software Law Review. http://www.ifosslr.org. Last accessed June 2013.

[11] P. Kruchten, “Architectural Blueprints — The “4+1” View Model of Software Architecture, “ IEEE Software 12 (6), November. 1995, pp. 42–50.

[12] lchecker A License Compliance Checker. http://code.google.com/p/lchecker/. Last accessed June 2013.

[13] F-F. Lidia and A. Vallecillo-Moreno, “An introduction to UML profiles”, UML and Model Engineering, vol. V, no. 2, April. 2004, pp. 6–13.

[14] Linux Foundation. Dependency Checker Tool http://www.linuxfoundation.org/sites/main/files/publications/lf_foss_compliance_dct.pdf. Last accessed June 2013.

[15] A. Lokhman, A. Luoto, S. Abdul-Rahman, and I. Hammouda, “OSSLI: Architecture Level Management of Open Source Software Legality Concerns, “ Proc. OSS 2012, 2012, pp. 356–361.

[16] B. Malcolm, “Software Interactions and the GNU General Public License, “ IFOSS L. Rev, 2(2), 2010, pp. 165–180.

[17] Ninka, a License Identification Tool for Source Code. http://ninka.turingmachine.org/. Last accessed June 2013.

[18] Open Source Initiative. http://www.opensource.org. Last accessed June 2013.

[19] OSLC, Open Source License Checker. http://sourceforge.net/projects/oslc. Last accessed June 2013.

[20] Sourceforge.net. http://sourceforget.net/. Last accessed June 2013.

[21] OSSLI project. http://ossli.cs.tut.fi/. Last accsedd June 2013.

[22] Papyrus. http://www.eclipse.org/modeling/mdt/papyrus/. Last accessed June 2013.

[23] Qualipso project. http://www.qualipso.org/licenses-champion. Last accessed June 2013.

[24] O. Räihä, Hadaytullah, K. Koskimies, and E. Mäkinen, “Synthesizing Architecture from Requirements: A Genetic Approach, ” Relating Software Requirements and Architecture (eds. P. Avgeriou, J. Grundy, J. G. Hall, P. Lago, and I. Mistrik), Chapter 18, Springer, 2011, pp. 307–331.

[25] Software Package Data Exchange (SPDX). http://spdx.org/. Last accessed June 2013.

[26] Systems and Software Engineering – Architecture Description. ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000), 2011, pp. 1–46.

[27] T. Tuunanen, J. Koskinen, and T. Kärkkäinen, “Automated Software License Analysis, “ Automated Software Engineering 16 (3-4), December. 2009, pp. 455–490.

[28] M. von Willebrand and M. P. Partanen, “Package Review as a Part of Free and Open Source Software Compliance, ” IFOSS L. Rev, 2(2), 2010, pp. 39–60.

[29] AGPL, Gnu Affero General Public License. http://www.gnu.org/licenses/agpl-3.0.html. Last accessed September 2013.

Can Business Process Management Benefit from Service Journey Modelling Language?

Eunji Lee, Amela Karahasanović
SINTEF ICT
Oslo, Norway
eunji.lee@sintef.no, amela@sintef.no

***Abstract*—Business process management aims to align the business processes of an organisation with customers' needs. Doing this is of particular importance for services and requires a good understanding of interactions among the stakeholders involved in service provision and consumption. Several business modelling languages have been proposed, such as Business Process Modelling Notation (BPMN), Business Process Executable Language (BPEL) and Web Services Choreography Description Language (WS-CDL). Although these languages provide good support for process modelling, their consideration of the customer's point of view seems to be insufficient. On the other hand, visualisations of customer journeys for the purpose of conceptualisation of new services have been successfully used in the area of service design. Our hypothesis is that a visual language presenting the customer journey through a service might be useful for aligning business processes of service providers with customers' needs and, in turn, contribute to the delivery of better services. We propose Service Journey Modelling Language (SJML) and report our first experience with it.**

Keywords-software engineering; business process management; services; visual languages

I. Introduction

Services play an important role in the global economy [1]. This heightens the need to understand business process management (BPM) in the context of services. BPM is used to improve the outcomes and operational agility of business performance by linking people, information flows, system and other assets in order to create and deliver value to customers [2].

Several standard languages have been used for business process management. However, the concept of the customer's perspective is not sufficiently considered enough in most of the languages.

Service providers need appropriate methods and languages to describe the entire service process from the customer's point of view. This employs knowledge from the areas of information visualisation, business process modelling languages, and service design.

The rest of the paper is organised as follows: Section II describes related work. Section III describes visual language for modelling service journey. Section IV concludes the paper and proposes future work.

II. Related Work

Information visualisation increases human cognition [3]. It helps people to more easily understand complex information [4], and changes over time that could otherwise be difficult to comprehend [5].

Moody drew visual information transmission with two processes: encoding and decoding [6]. We aim to develop a visual language for presenting customer journey through services (we call it Service Journey Modelling Language, or SJML) that includes graphical syntax and information about encoder, decoder and channels for effective communication, as presented in Figure 1.

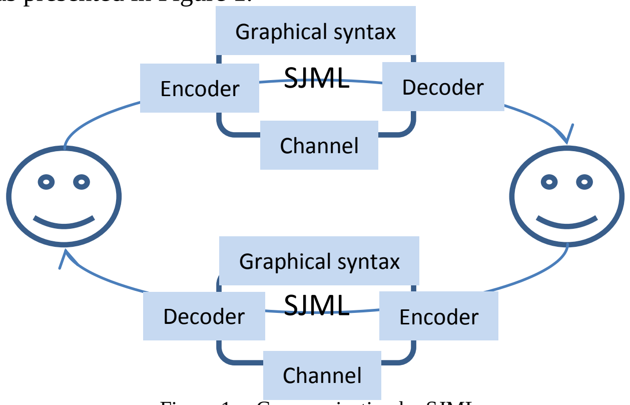


Figure 1. Communication by SJML

Several languages have been used for BPM, such as:

- **Business Process Modelling Notation (BPMN).** BPMN is a graphical notation that shows the steps in a business process and depicts a flow chart that defines business process workflows [7].
- **Business Process Execution Language (BPEL).** BPEL indicates a language that is used to define and execute business processes by using the interfaces via web services in order to export and import business information [8].
- **Web Services Choreography Description Language (WS-CDL).** WS-CDL is a non-executable language based on XML that enables global business processes to be shown [9].
- **ServiceML.** ServiceML includes three packages; Business-SoaML, Light-USDL and Service Journey Map, which consists of touchpoints. Touchpoint means a contact point or interaction

between a customer and a service provider during service delivery. The colours of the touchpoints stand for the type of behaviours (normal, ad-hoc and unexpected) and customer emotional stage (unhappy, neutral and happy).

Customer Journey Map (CJM) is a tool used in service design to visualise users' experience. A map is constructed with touchpoints. The details of service interactions and the associated emotions can be described in a highly accessible manner by using a CJM [10]. People use the map to see the service delivery process from the user's perspective. The CJM overview shows problem areas and opportunities for innovation, and the touchpoints assist in further analysis [10]. By using CJM, people can easily and quickly compare a service with its competitors [10].

Service blueprint is a technique that was introduced by Shostack and has been used in business and marketing [11]. It shows the series of service actions and the time flows related to the roles of stakeholders during service delivery by dividing them into front tasks and back tasks. Service blueprints enable managers to understand the entire process properly and provide useful information for new service development and its evaluation.

Above described languages and tools support modelling business processes. However there is a lack of methods for precise specification of concepts where the customer has a role as a co-producer [12]. While partly addressing this, CJM and service blueprints are mainly focusing on the conceptualisation and evaluation phases [13, 14]

III. VISUAL LANGUAGE FOR MODELLING SERVICE JOURNEY

The main goal of this research is to introduce a generic visual language which enhances the service design development/improvement process. We propose a visual language, called Service Journey Modelling Language (SJML) that supports aligning business processes of service providers with customers' needs. The language will be developed and evaluated in an iterative manner. Information visualisation theory [15] and communication theory [6] will form a theoretical basis for language design and evaluation. Information visualisation involves users, tasks and visualisation forms [15]. We aim to investigate which visualisation forms might improve communication between stakeholders when performing different tasks within service design and development.

As the first phase of our research, we wanted to investigate the needs of practitioners when designing new or improving existing services. How do different stakeholders such as designers, service developers and managers communicate with each other? Which information about customers and their interaction with services is essential when aligning business process with customers' needs? We developed the first version of the language and evaluated it in a half-day workshop with twenty-six employees of a university library.

A. Scope

Services can be divided into the following four areas according to the nature of the service act and the recipient of the service: services directed at peoples' bodies, services directed at physical possessions, services directed at peoples' minds and services directed at intangible assets [16]. Figure 2 gives examples of these services. We intend to use SJML within all four service areas.

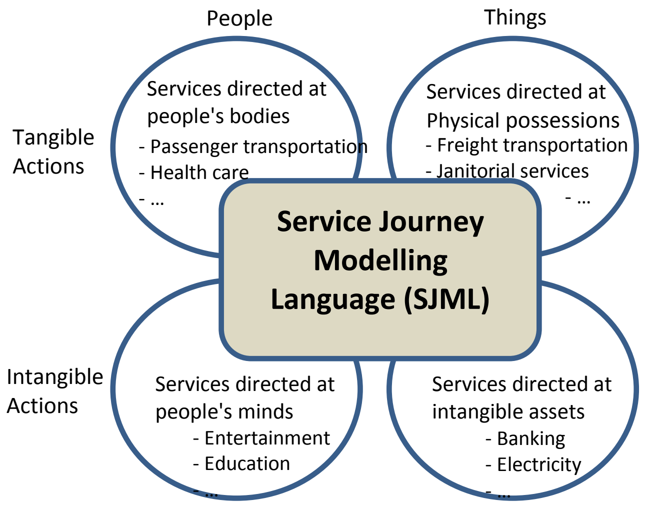

People
Things
Tangible Actions
Intangible Actions
Services directed at people's bodies
- Passenger transportation
- Health care
- ...
Services directed at Physical possessions
- Freight transportation
- Janitorial services
- ...
Service Journey Modelling Language (SJML)
Services directed at people's minds
- Entertainment
- Education
- ...
Services directed at intangible assets
- Banking
- Electricity
- ...

Figure 2. Service areas covered by SJML

Table I compares SJML with other similar languages. The second column gives the application domain of the language. BPMN, BPEL, WS-CDL and ServiceML are used in business process management, whereas SJML is used in service design and development. The second column indicates whether the language considers service providers' and/or customers' point of view. The third column indicates communication coverage. Front-communication means that the languages cover only communication between employees inside organisations. Front-end communication means that the language covers also communication between service providers and customers. ServiceML describes service experience from both customer's and service provider's view and cover communication between service providers and customers. However, ServiceML cannot display third party stakeholder. SJML aims to be customer-oriented, cover front-end communication and enables to illustrate existing and newly designed services.

TABLE I. COMPARISON OF THE LANGUAGES

Language	Domain	Perspective	Communication coverage
BPMN, BPEL, WS-CDL	Business	Service provider–oriented	Front communication
ServiceML	Business	Service provider /customer-oriented	Front-end communication
SJML	Service design	Customer-oriented	Front-end communication

Service design includes the following phases: ideation, conceptualisation, design, prototyping, development, implementation, evaluation, maintenance and improvement. Whereas CJM and Service blueprints support ideation and conceptualisation phase, SJML is expected to be used in the whole service design process from conceptualisation to improvement aiming to enhance service quality.

SJML aims to improve:

- communication to strengthen customer orientation and facilitate collaboration between all involved stakeholders through a common vocabulary and extensive use of visualisation;
- support for design and development, where language serves as a tool for managing the development and implementation of innovative service concepts; and
- support for the analysis of existing and new services to ensure consistency and overall customer experience across touchpoints and throughout the service life cycle.

B. Specification of SJML

Meetings, seminars, e-mail and telephone conversations, were used to specify requirements and to collect relevant data. From this requirement and data, functional requirement are specified like below.

- Source of requirement: internal, external meetings, seminars, e-mail and telephone conversation.
- Functional requirement:

Touchpoint. SJML consists of a sequence of touchpoints. Each touchpoint include symbols which show channels and devices that are used for the touchpoint.

Actor. The colour of the boundary indicates the actor initiating the touchpoint.

Status. The boundary style indicates the status of the touchpoint (solid boundary: completed, dashed boundary: missing and crossed touchpoint: failed).

The first version of SJML (SJML v1.0.) consists of terminology, symbols and model journey. Customer journey relevant terminology such as service and stakeholders was studied and summarized for better understanding. Symbols and visual elements were developed together with syntax and context. Visual symbols represent actions, devices and mediums that are used during service delivery process. We used SJML v1.0. for model service analysis of four different services (Going to the movies, Tax reporting, Retail purchase and Air travel). Both expected and actual journeys were mapped.

C. Evaluation and Results

A service design seminar was held at the science library at the University of Oslo in June 2013. The seminar consisted of a lecture about service design and two practical sessions. SJML was introduced and tested during one of these sessions. The session included a short introduction of SJML, eight tasks and discussion. Twenty-six librarians participated, and the entire session took about 30 minutes.

Participants were divided into four working groups and asked to make customer journey maps of the service process of borrowing paper and electronic books at the library using SJML. One blank icon plus seventeen book loan service relevant icons which were selected among 32 SJML icons were given to each group as a set (Figure 3). In this workshop, the actor and status concepts were not adapted.

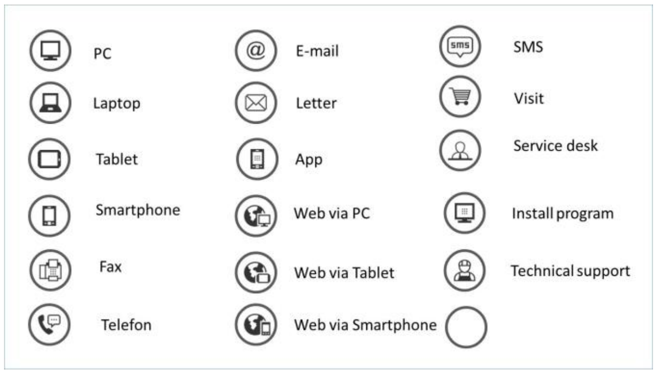


Figure 3. SJML icons given at the workshop

First task was to present customer journey for a customer borrowing a paper/electronic book (Figure 4.). The process includes extension of the loan and finishes with when the book is returned. Second task was to present customer journey for a customer ordering a paper/electronic book which the library does not have. The process includes extension of the loan and finishes when the book is returned. Participants were asked to make customer journey maps for the both existing (Figure 4.) and desired book loan service.

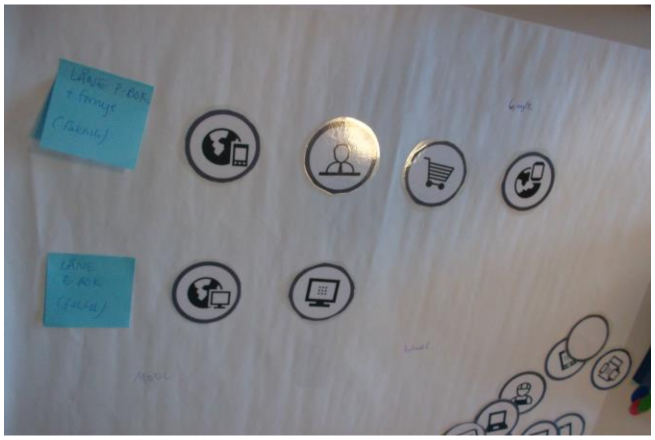


Figure 4. Customer journey maps for a customer borrowing a paper book (up) /electronic book (down) in existing book loan service at the library

The participants had no problems in understanding of SJML and using its symbols. Participants were able to describe and explain the service journey using the given SJML icons. However some of the participants were confused about using the symbols that look similar such as the icon of PC and the icon of web service via PC.

Participants also wanted to draw loops that happen repeatedly. However they did not know how to present this. It was also found that more icons were needed to illustrate library service specific touchpoints. These challenges will be addressed in the next version of SJML.

IV. Conclusion and Future Work

Services usually have a complex structure with several stakeholders, and their interests are intertwined. Aligning the business processes of an organisation with customers' needs is important for business process management, especially in service field. A good understanding of interactions among the stakeholders involved in service provision and consumption is need for this. Several modelling languages were introduced for business process management and several methods were suggested to support service design process. However, there is a lack of support for describing, communicating and analysing service concepts for stakeholders in a detailed way in order to develop and implement new services and improve existing services.

SJMLv1.0. was developed and tested by adapting information visualisation and visual communication theories together with requirements. We expect that SJML and associated methods can improve business process modelling by alleviating communication problems among different stakeholders.

We are going to develop and evaluate several versions of SJML in an iterative manner. They will be evaluated in collaboration with our industrial partners on real-life services they are providing. A literature review, interviews, prototyping, usability testing, post-mortem analysis and a living lab would be used in further research.

Acknowledgment

This study is part of the VISUAL project (2012–2016, project number 219606) funded by the Research Council of Norway and industrial partner companies, and involving Sintef ICT and the University of Linköping.

Thanks to Ragnhild Halvorsrud and Ida Maria Haugstveit for the fruitful collaboration in developing the first version of SJML, to Alma Culén and Andrea Gasparini for organising the workshop and to all participants who took part in the workshop.

References

[1] T. P. Soubbotina and K. Sheram, Beyond economic growth: Meeting the challenges of global development. World Bank Publications, 2000.

[2] Gartner, Inc., “Gartner IT Glossary - Business process management (BPM),” 2013. [Online]. Available: http://blogs.gartner.com/it-glossary/business-process-management-bpm-2/. [Accessed: 22-Aug-2013].

[3] S. K. Card, J. D. Mackinlay, and B. Shneiderman, Readings in information visualization: using vision to think. Morgan Kaufmann, 1999.

[4] Y. Rogers, H. Sharp, and J. Preece, Interaction Design: Beyond Human Computer Interaction (3rd edition). Wiley, 2011.

[5] C. Ware, Information visualization, vol. 2. Morgan Kaufmann, 2000.

[6] D. Moody, “The ‘physics’ of notations: toward a scientific basis for constructing visual notations in software engineering,” Softw. Eng. IEEE Trans., vol. 35, no. 6, 2009, pp. 756–779.

[7] The Object Management Group, “Business Process Management Initiative,” 2013. [Online]. Available: http://www.omg.org/bpmn/Documents/FAQ.htm. [Accessed: 12-Jun-2013].

[8] M. B. Juric, “A Hands-on Introduction to BPEL.” [Online]. Available: http://www.oracle.com/technetwork/articles/matjaz-bpel1-090575.html. [Accessed: 22-Aug-2013].

[9] Business Process Modeling Forum, “Business Process Modeling FAQ | BPM Forum,” 2013. [Online]. Available: http://www.bpmodeling.com/faq/. [Accessed: 12-Jun-2013].

[10] M. Stickdorn and J. Schneider, This is service design thinking. Wiley, 2010.

[11] G. L. Shostack, “Designing services that deliver,” Harv. Bus. Rev., vol. 62, no. 1, 1984, pp. 133–139.

[12] L. Witell, P. Kristensson, A. Gustafsson, and M. Löfgren, “Idea generation: customer co-creation versus traditional market research techniques,” J. Serv. Manag., vol. 22, no. 2, 2011, pp. 140–159.

[13] C. Tollestrup, “Conceptualising services: developing service concepts through AT-ONE,” Nord. Conf. Serv. Des. Serv. Innov., 2009, pp. 187–199.

[14] G. L. Shostack, “Service positioning through structural change,” J. Mark., 1987, pp. 34–43.

[15] C. Chen and Y. Yu, “Empirical studies of information visualization: a meta-analysis,” Int. J. Hum.-Comput. Stud., vol. 53, no. 5, 2000, pp. 851–866.

[16] C. H. Lovelock, “Classifying services to gain strategic marketing insights,” J. Mark., 1983, pp. 9–20.

A Method of Generation of Scenarios using Differential Scenario

Eiji Shiota
Graduate School of Science and Engineering
Ritsumeikan University
Kusatsu, Shiga, Japan
e-mail: shiota@cs.ritsumei.ac.jp

Atsushi Ohnishi
Department of Computer Science
Ritsumeikan University
Kusatsu, Shiga, Japan
e-mail: ohnishi@cs.ritsumei.ac.jp

***Abstract*—In scenario-based requirements engineering, system behaviours can be given by scenarios. First, we give a normal scenario of a system to be developed. Secondly, we can retrieve scenarios of similar behavior with the given scenario using differential information between the given scenario and a retrieved scenario. Thirdly, we retrieve alternative scenarios and exceptional scenarios of the retrieved scenario. Lastly, we can generate alternative scenarios and exceptional scenarios of the given scenario using the differential information. Our method will be illustrated with examples. This paper describes (1) a language for describing scenarios based on a simple case grammar of actions, (2) introduction of the differential scenario, (3) method and examples of scenario retrieval using the differential scenario and (4) method and example of scenario generation using the differential scenario. The effectiveness of the method is shown through an experiment.**

Keywords*—*scenario generation; scenario retrieval; differential scnenario; scenario-based requirements engineering

I. Introduction

Scenarios are important in software development [6], particularly in requirements engineering by providing concrete system description [16], [18]. Especially, scenarios are useful in defining system behaviors by system developers and validating the requirements by customers. In scenario-based software development, incorrect scenarios will have a negative impact on the overall system development process. However, scenarios are usually informal and it is difficult to verify the correctness of them. Errors in incorrect scenarios may include (1) vague representations, (2) lack of necessary events, (3) extra events, and (4) wrong sequence among events.

The authors have developed a scenario language named SCEL (SCEnario Language) for describing scenarios in which simple action traces are embellished to include typed frames based on a simple case grammar of actions and for describing the sequence among events[17], [19]. Since this language is a controlled language, the vagueness of the scenario written with SCEL language can be reduced. Furthermore, a scenario with SCEL can be transformed into internal representation. In the transformation, both the lack of cases and the illegal usage of noun types can be detected, and concrete words will be assigned to pronouns and omitted indispensable cases [14]. As a result, the scenario with SCEL can avoid the errors typed 1 previously mentioned.

Scenarios can be classified into (1) normal scenarios, (2) alternative scenarios, and (3) exceptional scenarios. A normal one represents the normal and typical behavior of the target system, while an alternative one represents normal but alternative behavior of the system and an exceptional one represents abnormal behavior of the system. There are many normal scenarios for a certain system. For example, a normal scenario represents withdrawal of a banking system, another normal scenario represents money deposit, another one represents wire transfer, and so on. Each normal scenario has several alternative scenarios and exceptional scenarios. In order to grasp all behaviors of the system, not only normal scenarios, but also alternative/ exceptional scenarios should be specified. However, it is difficult to hit upon alternative scenarios and exceptional scenarios, whereas it is easy to think of normal scenarios.

This paper focuses on automatic generation of alternative/exceptional scenarios from normal scenarios of a new software system to be developed. We adopt the SCEL language for writing scenarios, because the SCEL is a controlled language and it is easy to analyze scenarios written with the SCEL.

The paper is organized as follows. The SEL language is described in Section II. After that, differential scnario information is presented in Section III. Section IV and V describes scenario retrieval and scenario generation, respectively. Then Section VI provides an experiment for evaluation our method. Section VII discusses related researches and compares with our work. Lastly, Section VIII arrives at a conclusion.

II. Scenario language

A. Outline

The SCEL language has already been introduced [19]. In this paper, a brief description of this language will be given for convenience. A scenario can be regarded as a sequence of events. Events are behaviors employed by users or the system for accomplishing their goals. We assume that each event has just one verb, and that each verb has its own case structure [9]. The scenario language has been developed based on this concept. Verbs and their own case structures depend on problem domains, but the roles of cases are independent of problem domains. The roles include agent, object, recipient, instrument, source, etc. [9], [14]. Verbs and their case structures are provided in a dictionary of verbs. If a scenario describer needs to use a new verb, he can use it by adding the verb and its case structure in the dictionary.

We adopt a requirements frame in which verbs and their own case structures are specified. The requirements frame depends on problem domains. Each action has its case structure, and each event can be automatically transformed into internal representation based on the frame. In the transformation, concrete words will be assigned to pronouns and omitted indispensable cases. With Requirements Frame, we can detect both the lack of cases and the illegal usage of noun types [14].

We assume four kinds of time sequences among events: 1) sequential, 2) selective, 3) iterative, and 4) parallel. Actually most events are sequential events. Our scenario language defines the semantic of verbs with their case structure. For example, data flow verb has source, goal, agent, and instrument cases.

Suppose a scenario of purchasing a train ticket. One scenario may consist of just one event of buying a train ticket. Another scenario may consists of several events, such as 1) informing date, destination, and the number of passengers, class of cars, 2) retrieving train data base, 3) issuing a ticket, 4) charging ticket fee to a credit card, and so on. If the abstract levels of scenarios are different, it is quite difficult to correctly compare and analyze events of scenarios. SCEL language for writing scenarios solves this problem, because SCEL provides a limited actions and their case structure as described in Section 2-C, and scenarios with SCEL keep a certain abstract level of actions.

[Title: Reservation of a hotel room]
[Viewpoints: user, reservation system]
1.A user enters his membership number and his name to the reservation system.
2.The system validates the user with the membership number and the name.
3.The user enters retrieval information to the system.
4.The system retrieves available hotels from the database using the information.
5.The system shows available hotels to the user.
6.The user selects a hotel from the available hotels.
7. The system shows the room rate to the user.
8.The user enters the credit card number to the system.
9.The system asks the status of the card to a credit card company using the card number.
10.The system shows the reservation number to the user.

Fig. 1. Scenario example.

B. Scenario example

Fig. 1 shows a scenario of reservation of a hotel room written with our scenario language, SCEL. A title of the scenario is given at the first line of the scenario in Fig. 1. Viewpoints of the scenario are specified at the second line. In this paper, viewpoints mean active objects such as human, system appearing in the scenario. There exist two viewpoints, namely "user" and "reservation system." The order of the specified viewpoints means the priority of the viewpoints. In this example, the first prior object is "user," and the second is "reservation system." In such a case, the prior object becomes a subject of an event.

In this scenario, all of the events are sequential. Actually, event number is for reader's convenience and not necessary.

C. Analysis of events

Each event is automatically transformed into internal representation. For example, the 1st event "A user enters his membership number and his name to the reservation system" can be transformed into internal representation shown in Table I. In this event, the verb "enter" corresponds to the concept "data flow." The data flow concept has its own case structure with four cases, namely to say, source case, goal case, object case and instrument case. Sender corresponds to the source

TABLE I. INTERNAL REPRESENTATION OF THE 1ST EVENT.

Concept: Data Flow

source	goal	object	instrument
user	reservation system	membership number and name	*NOT specified*

case and receiver corresponds to the goal case. Data transferred from source case to goal case corresponds to the object case. Device for sending data corresponds to the instrument case. In this event, "membership number and name" correspond to the object case and "user" corresponds to the source case.

The internal representation is independent of surface representation of the event. Suppose other representations of the event, "the reservation system receives user's membership number and his name from a user" and "User's membership number and his name are sent to the reservation system by a user." These events are syntactically different but semantically same as the 1st event. These two events can be automatically transformed into the same internal representations as shown in Table I.

III. DIFFERENTIAL SCENARIOS

Systems that are designed for a similar purpose (e.g. reservation, shopping, authentication, etc) often have similar behaviors. Besides, if such systems belong to the same domain, actors and data resemble each other. In other words, normal scenarios of a similar purpose belonging to the same domain resemble each other. Since our scenario language provides limited vocabulary and limited grammar, the abstraction level of any scenarios becomes almost the same.

For one system, there exist several normal scenarios. In the case of ticket reservation, reservation can be written as a normal scenario and cancellation can be written as another normal scenario. For a certain normal scenario, there are several exceptional scenarios and alternative scenarios. To make a differential scenario, we select two normal scenarios of two different systems. Each of the two scenarios should represent almost the same purpose, such as reservation of some item.

The differential scenario consists of (1) a list of not corresponding words, (2) a list of not corresponding events, that is, deleted events which appear in one scenario (say, scenario A) and do not appear in the other (say, scenario B) and added events which do not appear in scenario A and appear in scenario B. We also provide (3) a list of corresponding words and (4) a list of corresponding events, and (5) a script to apply the above differential information for generating scenarios.

We generally assume that one to one correspondence between two nouns and one to one correspondence between two events. Fig. 2 shows a scenario of reservation of meeting room for residents in a city.

We compare the scenario of Fig. 1 with the scenario of Fig. 2 from top to bottom. First, we check the actors specified as viewpoints of the two scenarios. In the case of scenarios of Fig. 1 and 2, "user" in Fig. 1 corresponds to "citizen" in Fig. 2 and "reservation system" in Fig. 1 corresponds to "system" in Fig. 2. The correspondence should be confirmed by user.

Second, we check the action concepts of events. If there exist events whose action concept appears once in scenario A and B, respectively, we assume that these two events are

[Title: Reservation of a meeting room]
[Viewpoints: citizen, system]
1.A citizen enters reservation information to the system.
2. The system retrieves available room from the database using the information.
3.The system shows an available room to the citizen.
4. The citizen enters his name and telephone number to the system.
5.The system validates the citizen with the name and the telephone number.
6.The system shows the room rate to the citizen.
7.The citizen pays the rate to the system.
8.The system issues a receipt to the citizen.
9.The system shows the room number to the citizen.

Fig. 2. Normal scenario of reservation of a meeting room

TABLE II. THE INTERNAL REPRESENTATION OF THE FIRST FOUR EVENTS OF THE SCENARIO IN FIG. 1.

concept	**agent/ source**	**goal**	**object**
data flow	user	reservation system	membership number and name
validate	*system*	*user*	*membership number and name*
data flow	user	reservation system	retrieval information
retrieve	system	available hotels	database

probably corresponding to each other. For example, the concept of the 2nd event in Fig. 1 and the concept of the 5th event in Fig. 2 are "validate" and there are no more events whose concepts are "validate," so we regard these two events are probably corresponding to each other. Then we provide these two events to a user and the user will confirm that these two events are corresponding to each other by checking whether nouns of the same cases are corresponding or not.

If there exists an event whose action concept appears once in scenario A, but there exists two or more events of the action concept in scenario B, then we regard that one of the events of the concept in scenario B corresponds to the event in scenario A. So, we provide these events to system user and the user will check the corresponding events.

If there are two or more events whose concepts are same in two scenarios respectively, these events are candidates of corresponding events. Then we check that nouns of the same cases are corresponding to. Next we provide candidates to the user and he will select the corresponding event.

The first four events of the scenario in Fig. 1 can be transformed as shown in Table II. The internal representations of the first five events of the scenario in Fig. 2 are shown in Table III. In fact, the data flow concept has four cases, that is, source, goal, object, and instrument cases as shown in Table I, but the instrument cases are omitted in Table II and III for the space limitation.

For the 2nd event in Table II and the 5th event in Table III as shown with italic font, since the nouns of the cases of the two events are same or corresponding to each other, these two events are corresponding to each other. At this time we get "membership number and name" correspond to "name and telephone number." So, the 1st event in Fig. 1 corresponds to the 4th event in Fig. 2, because concepts are same and all of

TABLE III. THE INTERNAL REPRESENTATION OF THE FIRST FIVE EVENTS OF THE SCENARIO IN FIG. 2.

concept	**agent/ source**	**goal**	**object**
data flow	citizen	system	reservation information
retrieve	system	available room	database
data flow	system	citizen	available rooms
data flow	citizen	system	name and telephone number
validate	*system*	*citizen*	*name and telephone number*

TABLE IV. A LIST OF CORRESPONDING WORDS BETWEEN SCENARIO A AND SCENARIO B.

Nouns in scenario A	**Nouns in scenario B**
user	citizen
reservation system	system
membership number and name	name and telephone number
available hotels	available room
retrieval information	reservation information
reservation number	room number
hotel room	meeting room
hotels	room

the nouns of corresponding cases are corresponding to each other.

Similarly we detect corresponding events and corresponding nouns. Table IV shows a list of corresponding nouns. Fig. 3 shows corresponding events of the two scenarios. In Fig. 3, two events connected by an arrow are corresponding to each other. Events without an arrow have no corresponding events. The successive corresponding events are grouped into an event block. The first two events in Fig. 1 are grouped into a block named a1. The block a1 corresponds to a block named b2 consisting of the 4th and the 5th events in Fig. 2.

Finally, we can get the differential scenario between hotel reservation and meeting room reservation shown in Table IV, V, and VI and Fig. 3.

TABLE V. DELETED EVENTS FROM PERSPECTIVE SCENARIO A/ ADDED EVENTS FROM PERSPECT IVE SCENARIO B.

concept	**agent/ source**	**goal**	**object**
select	user	hotel	available hotels
data flow	user	system	credit card number
data flow	system	credit card company	credit card number

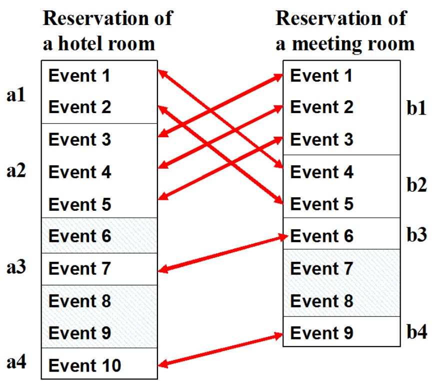


Fig. 3. Corresponding events.

TABLE VI. ADDED EVENTS FROM PERSPECTIVE SCENARIO A/ DELETED EVENTS FROM PERSPECTIVE SCENARIO B.

concept	agent/ source	goal	object
pay	citizen	system	room rate
data flow	system	citizen	receipt

```
1) change positions of block a1 and a2
2) delete events in Table V
3) insert events in Table VI followed by a4
4) change the corresponding nouns in Table IV
```

Fig. 4. Script applied to alternative/exceptional scenarios of scenario A.

IV. SCENARIO RETRIEVAL USING DIFFERENTIAL SCENARIO

In scenario-based software development, several scenarios should be specified. Since such scenarios may be revised, there exist a lot of scenarios of different revisions. When a scenario is given, it may be difficult to find similar scenarios or related scenarios to the given scenario by hand. We propose a retrieval method in order to get similar scenarios or related scenarios using the similar information of scenarios.

We assume that scenarios are analyzed based on the requirements frame in advance. As previously mentioned in Section 2, the requirements frame strongly depends on the problem domain. So, if case structures of verbs are different between two scenarios, we consider that these two scenarios are belonging to different domains each other. If all of the case structures are same, these scenarios can be classified into the same domain.

We propose two factors of the similarity between scenarios. One is related to same system. For example, a banking system provides several functions, withdrawal, deposit, loan, opening account, and so on. These functions are different each other, but both active objects, such as customer, bank clerk, ATM, banking system and inactive objects, such as bank card, cash, account in common appear in scenarios specifying behaviors of these functions of the banking system. The other factor is related to same or similar behavior. For example, behavior of train seat reservation and that of flight reservation are similar each other, although these systems are different.

A. Similarity of scenarios by system

If same nouns are used in scenarios, these scenarios probably specify behaviors of the same system. For example, "customer," "e-library system," and "librarian" appear in different scenarios, these scenarios can be regarded as scenarios of the same system. On the basis of the above discussion, we give an equation in order to measure the similarity of system of scenarios as below.

$$Similarity\ of\ system\ between\ two\ scenarios = \frac{the\ number\ of\ same\ nouns\ in\ events\ of\ the\ two\ scenarios}{the\ total\ number\ of\ nouns\ in\ events\ of\ the\ two\ scenarios} \quad (1)$$

As for scenarions in Fig.1 and 2, nouns in the events of these scenarios are shown in Table VII.

The total number of the nouns is 19 and the same nouns are "database", "name" and "room rate." So the similarity of system between these two scenarios becomes $\frac{3}{19}$.

TABLE VII. NOUNS IN THE EVENTS OF FIG.1 AND FIG.2

Scenario	nouns
Fig.1	available hotels(hotel), credit card company, credit card number (card number), database, membership number, name, retrieval information(information), reservation number, reservation system (system), room rate, status of the card, user
Fig2	available room, citizen, database, name, receipt, reservation information (information), room number, room rate(rate), system, telephone number

B. Similarity of scenarios by behavior

If scenario titles have a same verb, these scenario probably specify similar behaviors. For example, a scenario whose title is "a customer reserves a train seat" and another scenario whose title is "a user reserves a flight ticket" can be classified into similar scenarios from a behavioral viewpoint. However, a scenario whose title is "a customer purchases a train ticket" can be classified into similar scenarios with above ones. So, we think that scenarios are similar if titles of the scenarios have same verb, but this is not necessary.

Sequence of events in a scenario represents behaviors of users and system. If systems are different each other, nouns in events become different, even if events specify similar behaviors. So, we use corresponding events in the differential scenario. If two scenarios are similar each other from the viewpoint of behavior, the ratio of corresponding events becomes high.

On the basis of the above discussions, we give the second equation in order to measure the similarity of behaviors of scenarios as shown in below.

$$Similarity\ of\ behavior\ between\ the\ two\ scenarios = \frac{the\ number\ of\ corresponding\ events}{the\ total\ number\ of\ events\ of\ the\ two\ scenarios} \quad (2)$$

As shown in Fig.3, the total number of events is $10+9-7 = 12$ and the number of the same events is 7. So, the similarity of behavior between scenariosn of Fig.1 and 2 is $\frac{7}{12} = 0.58$ We consider that two scenarios whose similarity of behavior is greater than 0.5 are scenarios of similar behaviors.

In order to apply the differential information to another scenario of reservation of a hotel room, we also provide a script for application script shown in Fig. 4. Even if there exists a delete command in a script, event blocks will not be deleted when any event blocks in an applied scenario do not match with event blocks in the script. Even if there exists an insertion command in the script, event blocks will not be inserted when the following event block and the followed event block are missing in the applied scenario.

Fig. 5 shows the outline of the retrieval method of scenarios using the similar information of scenarios. We have been developing a prototype system based on the proposed method with C#.

C. Experiment

To evaluate our method, we compare the classification of scenarios by hands with the retrieval result by the method. Thirteen graduate students of CS department who well know both the scenario language and the problem domain classify nine scenarios for a standard scenario, while the same scenarios are also retrieved and classified based on the proposed

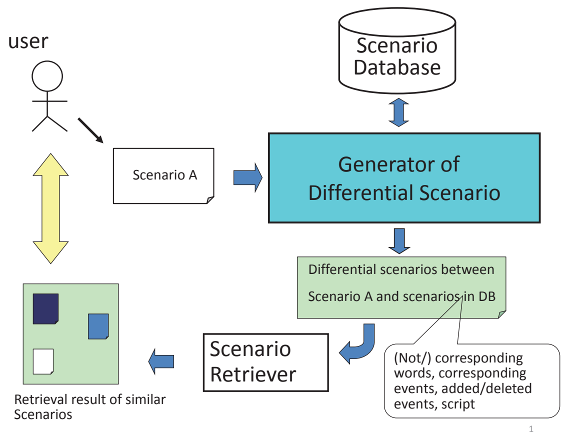

user
Scenario Database
Scenario A
Generator of Differential Scenario
Differential scenarios between Scenario A and scenarios in DB
(Not/) corresponding words, corresponding events, added/deleted events, script
Scenario Retriever
Retrieval result of similar Scenarios

Fig. 5. Outline of Scenario Retrieval.

TABLE VIII. SCENARIO CLASSIFICATION BY THE PROPOSED METHOD AND BY STUDENTS.

Scenario	Classification by method	Ratio of same result
Train ticket reservation	Different system, similar behavior	11/13
Flight ticket changing 1	Same system, different behavior	11/13
Flight ticket changing 2	Same system, different behavior	11/13
Train ticket reservation	Different system, similar behavior	11/13
Flight ticket reservation (Alternative scenario)	Same system, similar behavior	13/13
Bus ticket reservation	Different system, similar behavior	10/13
Claim for the loss on insurance	Different system, different behavior	13/13
Purchasing something	Different system, different behavior	12/13

method. A normal scenario of reservation of flight ticket was adopted as a standard scenario in this experiment. Table VIII shows the comparison of the scenario classifications.

In this experiment, nine scenarios are classified. The values of the column "Ratio of same result" mean the ratio of same classification between by our proposed method and by students. We investigated the reason why some students wrongly classified and found that they did not recognize the difference of systems correctly. After giving additional explanation of systems, the students adopted same classification of scenarios as classified by the proposed method. Through the experiment we confirm that our method can correctly classify scenarios for a given scenario and can retrieve similar scenarios with system/behavior.

V. SCENARIO GENERATION USING DIFFERENTIAL SCENARIO

Once the differential scenario between system A and B is given, we can apply it to another scenario of system A and get a new scenario of system B by changing corresponding words and by deleting or adding not-corresponding events. In this section, we apply the differential scenario described in the previous chapter to an alternative scenario of hotel reservation and get an alternative scenario of meeting room reservation [12].

A. *Examples of generation*

Fig. 6 shows an alternative scenario of hotel reservation. In this scenario, an aged user reserves a hotel room with a discount rate. By applying the differential scenario in Table IV, V, VI and Fig. 3 using the application script in Fig. 4, we can get a new alternative scenario of reservation of a meeting room for aged citizen as shown in Fig. 7. Lastly, the generated scenario is investigated by the user. He can modify the generated scenario to eliminate errors.

[Title: Reservation of a hotel room for aged users]
[Viewpoints: user, system]
1.A user enters his membership number and his name to the system.
2.The system validates the user with the membership number and the name.
3.The user enters retrieval information to the system.
4.The system retrieves available hotels from the database using the information.
5.The system shows available hotels to the user.
6.The user selects a hotel from the available hotels.
7.The system retrieves the date of birth of the user from the database using the membership number and the name.
8.The system checks the age of the user.
9.The system calculates the discount rate of the room for aged users.
10.The system shows the room rate to the user.
11.The user enters the credit card number to the system.
12.The system asks the status of the card to a credit card company using the card number.
13. The system shows the reservation number to the user.

Fig. 6. An alternative scenario.

[Title: Reservation of a meeting room for aged citizen]
[Viewpoints: citizen, reservation system]
1.The citizen enters reservation information to the system.
2.The system retrieves available room from the database using the information.
3.The system shows available room to the citizen.
4. The citizen enters his name and telephone number to the system.
5.The system validates the citizen with the name and the telephone number.
6.The system retrieves the date of birth of the citizen from the database using the name and the phone number.
7.The system checks the age of the citizen.
8.The system calculates the discount rate of the room for aged citizen.
9.The system shows the room rate to the citizen.
10.The citizen pays the rate to the system.
11.The system issues a receipt to the citizen.
12.The system shows the room number to the citizen.

Fig. 7. A generated new alternative scenario.

B. *Scenario generator using differential scenarios*

Fig. 8 shows the outline of the generation of scenarios using differential scenarios. We have been developing a prototype system based on the method. This system has been developed with C# on a Windows XP PC. The line of source code of the system is about 6,000. This system is a 4.5 man-month product.

This system mainly provides two functions. One is the derivation of the differential scenario between given two scenarios. The other is the application of the differential scenario to a specified scenario and the generation of a new scenario. If a user selects the former function and he specifies two scenarios, such as a scenario of the reservation of a hotel room and a scenario of the reservation of a meeting room, then differential scenario between them is derived.

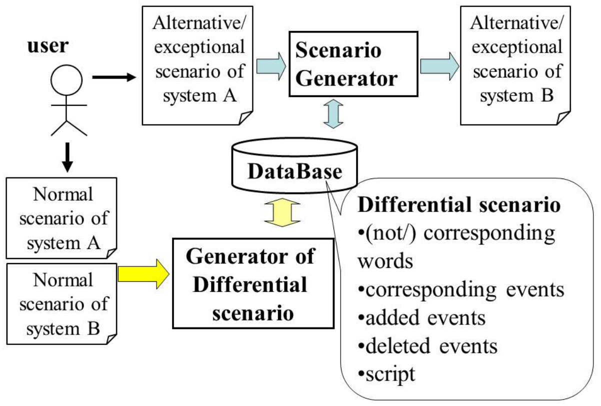
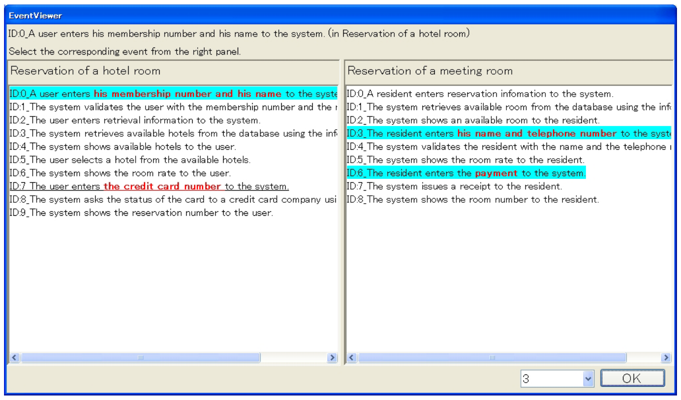

user
Alternative/ exceptional scenario of system A
Scenario Generator
Alternative/ exceptional scenario of system B
DataBase
Normal scenario of system A
Normal scenario of system B
Generator of Differential scenario
Differential scenario
•(not/) corresponding words
•corresponding events
•added events
•deleted events
•script

Fig. 8. Outline of scenario generation.


EventViewer

ID:0_A user enters his membership number and his name to the system. (in Reservation of a hotel room)

Select the corresponding event from the right panel.

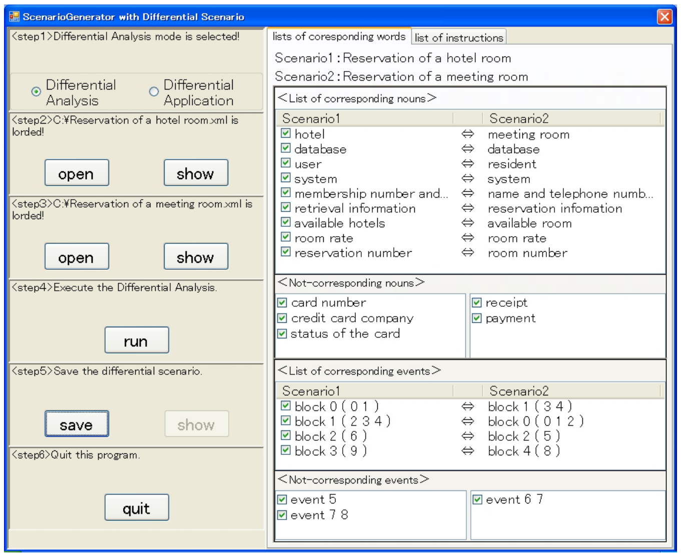
Reservation of a hotel room

ID:0_A user enters **his membership number and his name** to the syste
ID:1_The system validates the user with the membership number and the r
ID:2_The user enters retrieval information to the system.
ID:3_The system retrieves available hotels from the database using the inf
ID:4_The system shows available hotels to the user.
ID:5_The user selects a hotel from the available hotels.
ID:6_The system shows the room rate to the user.
ID:7_The user enters **the credit card number** to the system.
ID:8_The system asks the status of the card to a credit card company usi
ID:9_The system shows the reservation number to the user.

Reservation of a meeting room

ID:0_A resident enters reservation infomation to the system.
ID:1_The system retrieves available room from the database using the inf
ID:2_The system shows an available room to the resident.
ID:3_The resident enters **his name and telephone number** to the syste
ID:4_The system validates the resident with the name and the telephone r
ID:5_The system shows the room rate to the resident.
ID:6_The resident enters the **payment** to the system.
ID:7_The system issues a receipt to the resident.
ID:8_The system shows the room number to the resident.

Fig. 9. Candidates of corresponding events.


ScenarioGenerator with Differential Scenario

<step1>Differential Analysis mode is selected!

(● Differential Analysis) ○ Differential Application

<step2>C:¥Reservation of a hotel room.xml is lorded!

<step3>C:¥Reservation of a meeting room.xml is lorded!

<step4>Execute the Differential Analysis.

<step5>Save the differential scenario.

<step6>Quit this program.

Scenario1 : Reservation of a hotel room
Scenario2 : Reservation of a meeting room

<List of corresponding nouns>

Scenario1		Scenario2
☑ hotel	⇔	meeting room
☑ database	⇔	database
☑ user	⇔	resident
☑ system	⇔	system
☑ membership number and...	⇔	name and telephone numb...
☑ retrieval information	⇔	reservation infomation
☑ available hotels	⇔	available room
☑ room rate	⇔	room rate
☑ reservation number	⇔	room number

<Not-corresponding nouns>

☑ card number	☑ receipt
☑ credit card company	☑ payment
☑ status of the card	

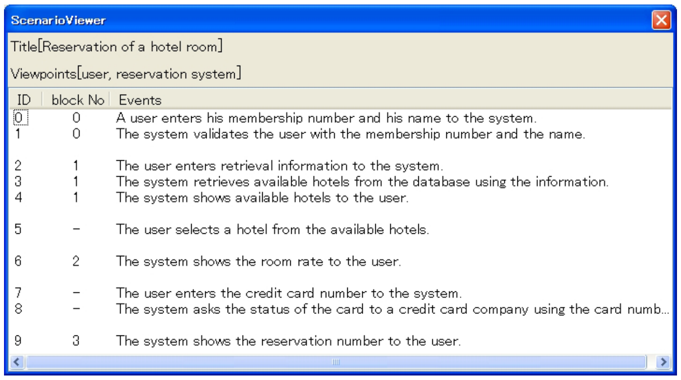
<List of corresponding events>

Scenario1		Scenario2
☑ block 0 (0 1)	⇔	block 1 (3 4)
☑ block 1 (2 3 4)	⇔	block 0 (0 1 2)
☑ block 2 (6)	⇔	block 2 (5)
☑ block 3 (9)	⇔	block 4 (8)

<Not-corresponding events>

☑ event 5	☑ event 6 7
☑ event 7 8	

Fig. 10. Derivation of a differential scenario.

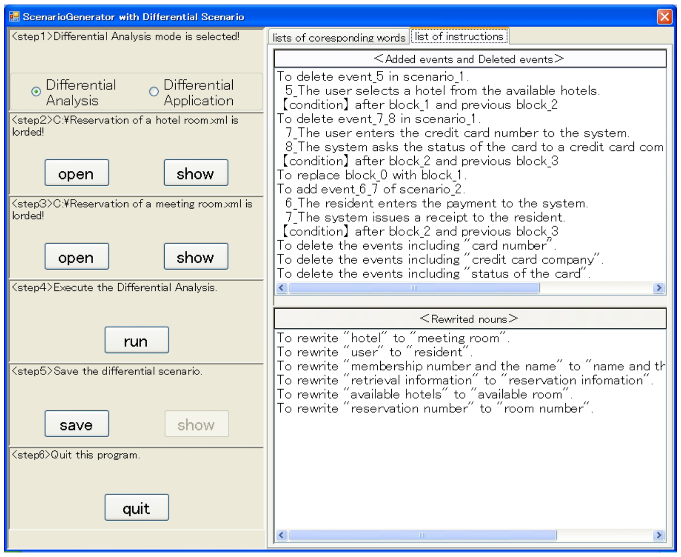

ScenarioViewer

Title[Reservation of a hotel room]

Viewpoints[user, reservation system]

ID	block No	Events
0	0	A user enters his membership number and his name to the system.
1	0	The system validates the user with the membership number and the name.
2	1	The user enters retrieval information to the system.
3	1	The system retrieves available hotels from the database using the information.
4	1	The system shows available hotels to the user.
5	–	The user selects a hotel from the available hotels.
6	2	The system shows the room rate to the user.
7	–	The user enters the credit card number to the system.
8	–	The system asks the status of the card to a credit card company using the card numb...
9	3	The system shows the reservation number to the user.

Fig. 11. Blocked events of the left scenario.


ScenarioGenerator with Differential Scenario

<step1>Differential Analysis mode is selected!

(● Differential Analysis) ○ Differential Application

<step2>C:¥Reservation of a hotel room.xml is lorded!

<step3>C:¥Reservation of a meeting room.xml is lorded!

<step4>Execute the Differential Analysis.

<step5>Save the differential scenario.

<step6>Quit this program.

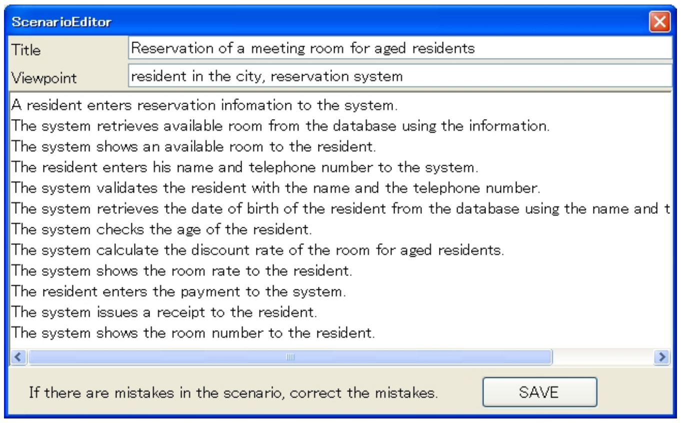
<Added events and Deleted events>

To delete event_5 in scenario_1.
 5_The user selects a hotel from the available hotels.
【condition】 after block_1 and previous block_2
To delete event_7_8 in scenario_1.
 7_The user enters the credit card number to the system.
 8_The system asks the status of the card to a credit card com
【condition】 after block_2 and previous block_3
To replace block_0 with block_1.
To add event_6_7 of scenario_2.
 6_The resident enters the payment to the system.
 7_The system issues a receipt to the resident.
【condition】 after block_2 and previous block_3
To delete the events including "card number".
To delete the events including "credit card company".
To delete the events including "status of the card".

<Rewrited nouns>

To rewrite "hotel" to "meeting room".
To rewrite "user" to "resident".
To rewrite "membership number and the name" to "name and th
To rewrite "retrieval information" to "reservation infomation".
To rewrite "available hotels" to "available room".
To rewrite "reservation number" to "room number".

Fig. 12. Generated script.


ScenarioEditor

Title	Reservation of a meeting room for aged residents
Viewpoint	resident in the city, reservation system

A resident enters reservation infomation to the system.
The system retrieves available room from the database using the information.
The system shows an available room to the resident.
The resident enters his name and telephone number to the system.
The system validates the resident with the name and the telephone number.
The system retrieves the date of birth of the resident from the database using the name and t
The system checks the age of the resident.
The system calculate the discount rate of the room for aged residents.
The system shows the room rate to the resident.
The resident enters the payment to the system.
The system issues a receipt to the resident.
The system shows the room number to the resident.

If there are mistakes in the scenario, correct the mistakes.

Fig. 13. Generated alternative scenario.

TABLE IX. SUBJECTS' ABILITIES OF SCENARIO ANALYSIS.

	Time(min.)	# of errors	# of events
A1	17	3	16
A2	17	0	19
A3	15	3	19
A4	13	2	17
B1	20	1	19
B2	19	0	19
B3	20	0	19
B4	20	0	19

TABLE X. SCENARIOS OF CD RENTAL SYSTEM.

id	title	the number of events
1	CD rental	19
2	CD rental failure by upper limitation	7
3	Return of CD	6
4	Retrun of CD with penalty	9
5	Retrieval of CDs	7
6	Registration of CDs	8
7	Registraion of a new member	16
8	Cancelation of a member	10

In Fig. 9, the user selects the corresponding event for the 1st event of the left-hand scenario. Two events are provided as candidates of corresponding events (the 4th event and the 7th event of the right-hand scenario). Since nouns with boldface font of the events are not registered in the list of corresponding words at that time, the user selects a corresponding event by specifying the id number of the event.

In this case, the user specifies the 4th event of the right-hand scenario as a corresponding event of the 1st event of the left-hand scenario by specifying the id number 3 in the bottom and right-side of the window in Fig. 9. The system automatically registers the correspondence between "membership number and name" of the left-hand scenario and "name and telephone number" of the right-hand scenario in the list of corresponding words. Likewise corresponding words and corresponding events will be determined and registered in the lists, respectively.

In Fig. 10, a list of corresponding words and a list of corresponding events are displayed in the right-hand side of the window.

In Fig. 11, events of the left-hand scenario in Fig. 9 are blocked. There are 4 blocks numbered 0, 1, 2 and 3 respectively. Three events are not blocked and they do not have any corresponding events.

In Fig. 12, an application script is displayed. By applying this script to an exceptional/alternative scenario of the reservation of a hotel room, an exceptional/alternative scenario of the reservation of a meeting room will be derived as shown in Fig. 13.

VI. EXPERIMENT

In order to evaluate our method and system, we performed an experiment. The purposes of the experiment are to confirm the following benefits.

1) to lessen elaboration of writing scenarios
2) to make a scenario of high quality

A. *Outline of the experiment*

Eight students who are graduate students belonging to software engineering laboratory, Ritsumeikan university are divided into two groups of four subjects that named group A and B. Prior to the experiment, we explained scenario language and the way of scenario writing for two hours. We chose a rental system as problem domain. We also gave a job description of a rental system to provide domain knowledge to subjects.

TABLE XI. RESULT OF THE EXPERIMENT.

Scenario id	Group A		Group B	
	Time(min.)	errors	Time(min.)	errors
1	-	-	-	-
2	4	0	10	0
3	1	0	7	0
4	2	0	15	1
5	3	0	10	0
6	8	0	7	0
7	2	0	14	6
8	1	0	5	0
average except for the scenario 1	3.0	0	9.7	1.0

Since the quality of generated scenarios depends on the ability of scenario writing and scenario analysis of subjects, we checked the ability of subjects prior to the experiment. We gave a normal scenario of borrowing a book at a library and asked to subjects to write a normal scenario of borrowing a CD at a CD rental shop. The result is shown in Table IX. A1, A2, A3, and A4 are members of group A, while B1, B2, B3, and B4 are members of group B. It took 17.6 minutes on average to write the scenario. The number of errors in a scenario of Group A is 2 on average, while the number of errors in a scenario of Group B is 0.5 on average. We confirmed that subjects' abilities of scenario writing and scenario analysis are different. The ability of Group A is less than that of Group B. This fact means that the quality of scenarios of Group A is usually less than that of Group B. We gave a correct scenario of borrowing a CD to all the members and pointed out the mistakes.

B. *Generation vs. description of scenarios*

We provided scenarios of a library system to the members of the two groups. These scenarios consist of 5 normal scenarios, and 2 exceptional scenarios. The member of group A wrote a normal scenario of borrowing a book and gets a differential scenario between scenario of borrowing a book and a scenario of borrowing a CD. Then they get the scenarios of CD rental system automatically generated using our proposed method and system, while the members of group B wrote one or two scenarios of the CD rental system by themselves using corresponding scenarios of the library system. We checked generated scenarios of group A and written scenarios of group B by comparing correct scenarios with them.

Table X shows a list of scenarios of the CD rental system prepared as correct scenarios by the authors. Scenario id number 3, 5, 6, 7 and 8 are normal scenarios of the CD rental system, while a scenario of no.2 and 4 are exceptional scenarios.

Table XI shows the result of experiment. It took extra 3.0 minutes on average to generate differential scenario for Group A. In using our method and system, scenarios are automatically generated, but the subjects need to check the generated scenarios. It took 3.0 minutes on average to check the scenarios. In checking none of the subjects found any errors in the generated scenarios. This means that our method and system generates exactly correct scenarios. In order to write scenarios by Group B, it took 9.7 minutes on average.

Actually, the ability of writing scenario of Group A is less than that of Group B, but the quality of generated scenarios by Group A is better than the quality of written scenarios by Group B as shown in Table XI. Through the experiment, we found that our method and system improve the correctness of the scenario and lessen the writing time.

VII. Related work

There is an obvious trend to define scenarios as textual description of the designed system behaviors. The growing number of practitioners demanding for more "informality" in the requirements engineering process seems to confirm this trend. Most of these papers describe how to use scenarios for the elicitation [15] or exploration [10] of requirements. The authors believe that it is also important to support both the generation and the classification of scenarios.

Ben Achour proposed guidance for correcting scenarios, based on a set of rules [1]. These rules aim at the clarification, completion and conceptualization of scenarios, and help the scenario author to improve the scenarios until an acceptable level in terms of the scenario models. Ben Achour's rules can only check whether the scenarios are well written according to the scenario models. We propose a method of generating exceptional scenarios and alternative scenarios from a normal scenario.

Neil Maiden et al. proposed classes of exceptions for use cases [11]. These classes are generic exceptions, permutations exceptions, permutation options, and problem exceptions. With these classes, alternative courses are generated. For communication actions, 5 problem exceptions are prepared, that is, human agents, machine agents, human-machine interactions, human-human communication, and machine-machine communication. They proposed a method of generating alternative paths for each normal sequence from exception types for events and generic requirements with abnormal patterns [3], [13], [15], [16]. Our approach for generating scenarios with a differential scenario is independent of problem domains.

Daniel Amyot et al. derive a scenario from use case map [2]. In order to generate several scenarios, they have to prepare several use case maps, while we have to prepare just one normal scenario with our approaches.

Christophe Damas et al. synthesize annotated behavior models from scenarios. They generate a state transition model from several scenarios and this model covers all scenario examples [7], [8]. However, they cannot generate scenarios of different systems, while our approach enables to generate scenarios of different systems.

Yu-Chin Cheng et al. proposes a generation method of attack scenarios [4]. Using attack patterns, attack state transition model, attack scenarios can be generated. Their approach focuses on just attack scenarios via network, but we provide a generation method of exceptional scenarios and alternative scenarios.

Dave Clarke et al. propose abstract delta modeling method to facilitate automated product derivation for software product lines. However, it seems difficult to give a correct delta model, while our approach enables to produce a correct differential scenario by giving two different scnarios.

VIII. Conclusion and future work

We have developed a frame base scenario language and a method of generating differential scenario between two scenarios. We have also developed a retrieval method of similar scenarios with system/behavior for a given scenario using the differential scenario and a generation method of alternative/exceptional scenarios for a given scenario using the differential scenario. The effectiveness of these two methods are validated through an experiment.

In order to retrieve more efficiently similar scenarios with differential scenario, using pre-conditions and post-conditions just like the selection of rules applicable to verify the correctness of scenarios [17] is left as our future work.

References

[1] C. B. Achour, "Guiding Scenario Authoring," Proc. 8th European-Japanese Conference on Information Modeling and Knowledge Bases, 1998, pp.181-200.

[2] D. Amyot, D. Y. Cho, X. He, and Y. He, "Generating Scenarios from Use Case Map Specifications," Proc. 3rd QSIC, Dallas, USA, 2003, pp.108-115.

[3] I. Alexander and N. A. M. Maiden, Scenarios, Stories, Use Cases, Through the Systems Development Life-Cycle, John Wiley & Sons, Ltd., 2004, pp.161-177.

[4] Y. C. Cheng, et al., "Generating Attack Scenarios with Causal Relationship," Proc. of IEEE International Conference on Granular Computing (GRC2007), 2007, pp.368-373.

[5] D. Clarke, M. Helvensteijn, and I. Schaefer, "Abstract delta modeling," Proc. 9th GPCE'10, 2010, pp.13-22.

[6] A. Cockburn, Writing Effective Use Cases, Addison-Wesley, USA, 2001

[7] C. Damas, B. Lambeau, P. Dupont, and A. Lamsweerde, "Generating Annotated Behavior Models from End-User Scenarios," IEEE Transactions on SE, Volume 31, Issue 12, 2005, pp.1056-1073.

[8] C. Damas, B. Lambeau, and A. Lamsweerde, "Scenarios, goals, and state machines, a win-win partnership for model synthesis," Foundations of Software Engineering, Proc. 14th ACM SIGSOFT international symposium on Foundations of Software Engineering, 2006, pp.197-207.

[9] C. J. Fillmore, The Case for Case, in Universals in Linguistic Theory, Holt, Rinehart and Winston, 1968.

[10] J. C. S. P. Leite, et.al., "Enhancing a Requirements Baseline with Scenarios," Proc. 3rd RE, 1997, pp.44-53.

[11] N. A. M. Maiden and M. Hare, "Problem Domain Categories in Requirements Engineering," International Journal of Human-Computer Studies, 49, 1998, pp.281-304.

[12] M. Makino and A. Ohnishi, "Scenario Generation Using Differentail Acenario Information," IEICE Trans. Information ans Systems, Vol.E95-D, No.4, pp.1044-1051.

[13] A. Mavin and N. A. M. Maiden, "Determining socio-technical systems requirements, experiences with generating and walking through scenarios," Proc. 11th IEEE RE, 2003, pp.213-222.

[14] A. Ohnishi, "Software Requirements Specification Database on Requirements Frame Model," Proc. IEEE 2nd ICRE, 1996, pp.221-228.

[15] A. G. Sutcliffe and M. Ryan, "Experience with SCRAM, a SCenario Requirements Analysis Method," Proc. 3rd ICRE, 1998, pp.164-171.

[16] A. G. Sutcliffe, N. A. M. Maiden, S. Minocha, and D. Manuel, "Supporting Scenario-Based Requirements Engineering," IEEE Trans. SE, Vol.24, No.12, 1998, pp.1072-1088.

[17] T. Toyama and A. Ohnishi, "Rule-based Verification of Scenarios with Pre-conditions and Post-conditions," Proc. 13th IEEE RE2005, 2005, pp.319-328.

[18] K. Weidenhaupt, K. Pohl, M. Jarke, and P. Haumer, "Scenarios in System Development, Current Practice," IEEE Software, March, 1998, pp.34-45.

[19] H. Zhang, A. Ohnishi, "Transformation between Scenarios from Different Viewpoints," IEICE Trans. Information and Systems, Vol.E87-D, No.4, 2004, pp.801-810.

Towards a UML Meta Model Extension for Aspect-Oriented Modeling

Meriem Chibani
Department of Mathematics and Computer Science
University of Oum El Bouaghi
Oum El Bouaghi, Algeria
e-mail: c.meriem@univ-oeb.dz

Brahim Belattar
Department of Computer Science
University of Batna
Batna, Algeria
e-mail: brahim.belattar@univ-batna.dz

Abdelhabib Bourouis
Department of Mathematics and Computer Science
University of Oum El Bouaghi
Oum El Bouaghi, Algeria
e-mail: a.bourouis@univ-oeb.dz

Abstract— **The aspect-oriented programming paradigm (AOP) as a way of improving the separation of concerns principle has emerged initially at the programming level using strong languages like AspectJ. Currently, it becomes mature to stretch at premature stages of the software development process namely, the Aspect-Oriented Software Development (AOSD) which is a popular topic of software engineering research that leads to more dependable, reusable and maintainable artifacts. In this paper, we propose a UML profile for modeling crosscutting concerns where the separation of concerns is maintained to the level of code and the weaving is done by an AspectJ compiler.**

Keywords-Aspect-Oriented Programming (AOP); UML profile; AspectJ; Aspect-Oriented Software Development.

I. INTRODUCTION

Besides functional concerns, software system development requires other concerns, namely crosscutting concerns as logging, distribution, error handling and security. These concerns cross cut the system functional modules, which produces a scattered and tangled design and decreases software's maintainability and modularity. The object-oriented paradigm does not satisfy the separation of concerns principle. It provides a powerful way to separate core concerns but it could not modularize crosscutting concerns in separate units. The aspect-orientation has originally emerged at the programming level with the well-known AspectJ language [1], in the late 1990s. Its use is no longer restricted to the programming level but more and more stretches over early phases of the software development life cycle such as requirements engineering, analysis and design. This new field is called the Aspect-Oriented Software Development (AOSD).

Aspect-oriented programming has emerged as a solution paradigm to overcome modularization problem. It distinguishes between the different categories of concerns, decreases coupling between concerns and more generally, it increases reuse. An AOP system may include many constructs where the central one is the aspect unit, which consists of two parts: dynamic crosscutting constructs and static ones. Dynamic crosscutting constructs provide a way to affect the behavior of a system. Join points are the points in the execution flow of an application; and pointcuts, a mechanism for selecting join points. The aspects have advices that are attached to one or more join points. When an advice is attached to join points, it will be executed, guided by its modifier which may specify the execution time relative to the join points: before, after, around, after exception or even after return value. These advices have an additional instance variable named thisJoinPoint that encapsulates the contextual information captured from the current junction. On the other hand, static crosscutting constructs alter static structure of the system. For example, when implementing tracing crosscutting concern, the introduction of a logger field into each traced class could be needed and inter-type declaration constructs make such modifications possible. In some situations, the need to detect certain conditions could arise, typically the existence of particular join points, before the execution of the system for which weave-time declaration constructs are suitable [2]. Furthermore, one of the main elements of AOP is the "weaving" mechanism which composes classes and aspects to produce a system with a new semantics. It could be performed before or after compilation and is known as static weaving. On the other hand, dynamic weaving is performed at load-time or run-time [3].

For an Aspect-Oriented Modeling (AOM) notation that provides a foundation for achieving better concern separation and integration, there is a need for several requirements. A general purpose, UML-based visual modeling language has several advantages over textual and domain specific alternatives. The notation should be complete, which means having a supporting abstraction for each of the commonly accepted AOSD concepts (aspect, component, pointcut, advice, static and dynamic crosscutting, Aspect-component relation and aspect-aspect relation). Furthermore, different concepts should be implicitly or explicitly mapped to different existing or new first-class UML elements. The notation should be independent from implementation language until the lowest level of detail is provided. In this way, the resulting aspect-oriented architectural models could be easily translated into elements of distinct aspect-oriented programming languages/frameworks and detailed design notations. Finally, the integrated UML-based notation should promote simplicity and avoid unnecessary extensions [4].

The Unified Modeling Language (UML) is a standard object oriented modeling language for specifying, visualizing, constructing, and documenting the artifacts of a system process. To enable it to represent the AOSD concepts at the design level, two alternatives are available. The general extension alternative aims at modifying the meta model of UML to include concepts related to the paradigm and is currently impractical due to a lack of tools support. The second alternative aims at building a UML profile which provides extension mechanisms [5]. UML extension mechanisms are based on "Stereotypes", "Tagged Values", and "Constraints" concepts. Briefly said, stereotypes are means of extending the UML metamodel classes, while tagged values are properties for stereotypes and constraints are used to restrict the stereotype vocabulary.

In this paper, we propose a UML v2.4 profile for modeling crosscutting concerns at the design level. The separation of concerns is maintained to the level of code and the weaving is done by an AspectJ compiler. We have used only UML class diagrams where the system behavior is not specified in UML behavioral diagrams.

The rest of the paper is organized as follows. Section 2 describes briefly the related work. Section 3 presents the proposed profile, while Section 4 discusses an application example. Finally, a conclusion is given in Section 5.

II. RELATED WORK

An aspect-oriented UML profile is one of the most challenges in closing the gap between AOP and aspect-oriented modeling phases. Initial discussion on UML profile was presented in [6], which proposed the specification of aspects as stereotypes on classes and aspects behavior as association relationship using collaboration diagram. The profile was specific for synchronization aspect and without addressing joinpoints, advice and pointcut concepts. It was later extended to include advice and pointcut specification in [7]. Similarly, in [8][9], initial aspect-oriented extensions using UML metamodels were described with a lack in graphical representation of most aspect-oriented constructs such as static crosscutting, join point and pointcuts.

In contrast to previous works, a complete AspectJ profile without textual specification was discussed by Evermann [10]. It was developed using the commercial tool MagicDraw with XMI (XML Metadata Interchange) format which allows easy code generation. However, it has inconsistencies compared to what is required by the paradigm and the proof was provided by a process for aspect-oriented profile checking in [11]. In [12], Evermann profile was extended to support aspect-oriented frameworks taking into consideration some AspectJ idioms, patterns and also stereotypes from a profile for object-oriented frameworks called UML-F.

In the terminology of Model Driven Architecture (MDA), unlike the previous works, which allow modeling only of Platform Specific Models (PSM), a Platform Independent Modeling (PIM) profile was developed in [13], after the identification of commonalities and differences between two representative AOSD implementations. As shown in Table 1, the significant differences between the implementation languages, i.e., AspectJ and AspectS, make the resulting profile complex to apply to models. Thus, a profile dedicated to a platform-specific technology is the candidate solution for reducing complexity [14].

TABLE I COMPARAISON OF AOP APPROACHES [14]

	AspectJ	**AspectS**	**AspectML**
Aspects can be instantiated	×	√	AspectML does not have an aspect construct.
Aspect inheritance	×	√	
Nested aspects	√	×	
Privileged aspects	√	×	
Polymorphic pointcuts	×	×	√
Polymorphic advices	×	×	√
Advice on field access	√	×	NA

Recently, Gowri [15] modeled joinpoint as sequence diagrams and it adopted XMI to deploy the profile in available CASE tools. It is a generic profile that captures only few of the AspectJ extensions.

The present proposal is an extension of the Evermann profile with several improvements. It represents a complete AspectJ imitation with two main contributions:

- Extending Evermann profile to comprise static crosscutting representation as shown in Figure 1, with highlighted stereotypes, e.g., the weave-time error and warning declarations constructs.
- Doing a considerable number of changes, for instance at the level of the used metaclasses and relations between stereotyped profile elements in order to eliminate Evermann profile complexity and improving efficiency, e.g., the metaclass Property is sufficient to represent the pointcut instead of the metaclass StructuralFeature, add the conditionalPointcut stereotype, etc.

III. THE PROPOSED PROFILE

Our profile is developed using the UML commercial tool MagicDraw [16]. It provides an efficient graphical UML2 editor for modeling and profiling with OCL verification engine for constraints checking.

A. *Aspect*

Aspect represents the modular unit in AOP paradigm that includes all crosscutting constructs such as advice and pointcut. The aspect is like a class, which may have both attributes and operations, access modifiers (public, private, protected or package), the ability to extend other classes, realize interfaces in addition to the fact that they may be abstract. Thereby, aspects are modeled by means of a stereotype <<aspect>> of Class, as shown in the Figure 1. Despite their similarities, aspects are different from classes and in order to overcome this, additional attributes and

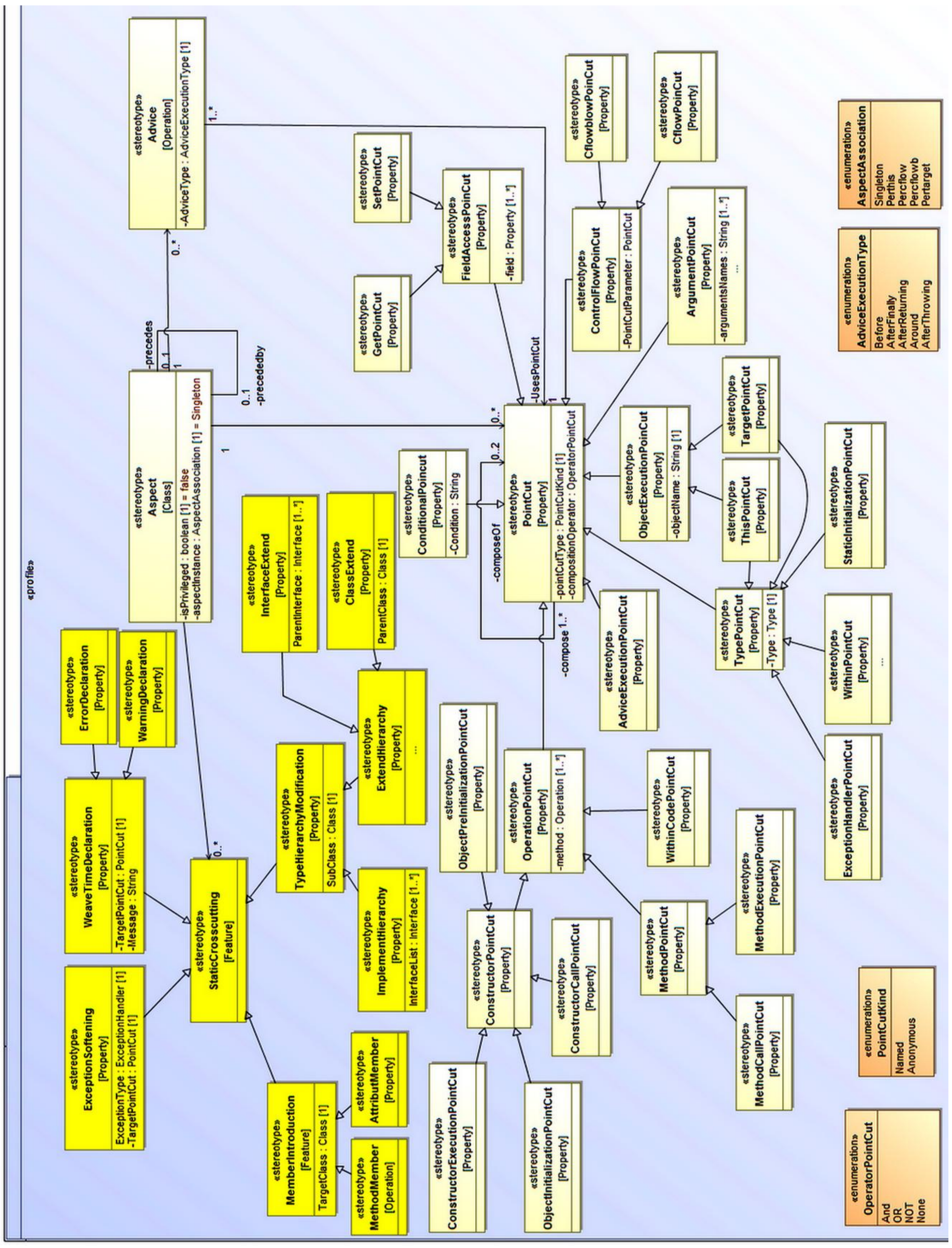

«profile»
«stereotype» Advice [Operation]
-AdviceType : AdviceExecutionType [1]
«stereotype» Aspect [Class]
-isPrivileged : boolean [1] = false
-aspectInstance : AspectAssociation [1] = Singleton
-precedes
-precededby
«stereotype» PointCut [Property]
-pointCutType : PointCutKind [1]
-compositionOperator : OperatorPointCut
-UsesPointCut
-composeOf
-compose 1..*
«stereotype» SetPointCut [Property]
«stereotype» GetPointCut [Property]
«stereotype» FieldAccessPoinCut [Property]
-field : Property [1..*]
«stereotype» CflowblowPoinCut [Property]
«stereotype» CflowPoinCut [Property]
«stereotype» ControlFlowPoinCut [Property]
-PointCutParameter : PointCut
«stereotype» ArgumentPointCut [Property]
-argumentsNames : String [1..*]
«stereotype» ConditionalPoincut [Property]
-Condition : String
«stereotype» ObjectExecutionPoinCut [Property]
-objectName : String [1]
«stereotype» TargetPointCut [Property]
«stereotype» ThisPointCut [Property]
«stereotype» StaticInitializationPointCut [Property]
«stereotype» TypePointCut [Property]
-Type : Type [1]
«stereotype» WithinPointCut [Property]
«stereotype» AdviceExecutionPointCut [Property]
«stereotype» ExceptionHandlerPointCut [Property]
«stereotype» InterfaceExtend [Property]
ParentInterface : Interface [1..*]
«stereotype» ClassExtend [Property]
ParentClass : Class [1]
«stereotype» ExtendHierarchy [Property]
«stereotype» TypeHierarchyModification [Property]
SubClass : Class [1]
«stereotype» ImplementHierarchy [Property]
InterfaceList : Interface [1..*]
«stereotype» ErrorDeclaration [Property]
«stereotype» WarningDeclaration [Property]
«stereotype» WeaveTimeDeclaration [Property]
-TargetPointCut : PointCut [1]
-Message : String
«stereotype» StaticCrosscutting [Feature]
«stereotype» ExceptionSoftening [Property]
ExceptionType : ExceptionHandler [1]
-TargetPointCut : PointCut [1]
«stereotype» MemberIntroduction [Feature]
TargetClass : Class [1]
«stereotype» AttributMember [Property]
«stereotype» MethodMember [Operation]
«stereotype» ObjectPreInitializationPointCut [Property]
«stereotype» OperationPointCut [Property]
-method : Operation [1..*]
«stereotype» WithinCodePointCut [Property]
«stereotype» ConstructorPointCut [Property]
«stereotype» ConstructorCallPointCut [Property]
«stereotype» MethodPointCut [Property]
«stereotype» MethodExecutionPointCut [Property]
«stereotype» MethodCallPointCut [Property]
«stereotype» ConstructorExecutionPointCut [Property]
«stereotype» ObjectInitializationPointCut [Property]
«enumeration» AspectAssociation
Singleton
Perthis
Percflow
Percflowb
Pertarget
«enumeration» AdviceExecutionType
Before
AfterFinally
AfterReturning
Around
AfterThrowing
«enumeration» PointCutKind
Named
Anonymous
«enumeration» OperatorPointCut
And
OR
NOT
None

Figure 1. AspectJ profile.

constraints on the metaclass Aspect are used.

1) *Attributes*

- isPriviliged: a Boolean which indicates if the aspect has a special privileged access specifier. If true, the aspect may access to private members of the classes which are crosscutting.
- aspectInstance: specifies the aspect instantiation model. Its possible values are: perthis, pertarget, percflow, singleton, or percflowb. Its default value is singleton, which means that the aspect has a unique instance.
- Precedence: it is modeled using a recursive (reflexive) association and determines the execution order of aspects with the same join point.

2) *Constraints*

In contrast to the class, concrete aspect could not declare generic parameters. Further, concrete aspect is not available for inheritance.

B. *Advice*

Advice is a dynamic construct in AspectJ, whereby it alters the behavior of the system at joinpoints selected by pointcuts. Because both advice and method express the behavior, have name, have arguments, could throw exceptions and have a body, we model advices using the metaclass Advice which extends the metaclass Operation.

1) *Attributes*

AdviceExecutionType: enumeration attribute that determines the type of the advice, i.e., before, after or around.

2) *Constraints*

In contrast to the method, which applies through an explicit call, the advice applies automatically in crosscutting manner. This is why an advice doesn't have an access specifier and only the "around" advice includes return type.

C. *Pointcut*

Pointcut selects the joinpoints with a structural description and has no relation with the dynamic behavior. This is why we model it using the metaclass Property and we add the constraint that the pointcut stereotype may be only applied to classes that are stereotyped Aspect. Furthermore, the metaclass Pointcut has additional attributes as follows:

- pointcutType: determines if the pointcut has a name or is anonymous.
- A pointcut may be composite, including other pointcuts using the OperatorPointcut enumeration. This mechanism is specified using a recursive association.

D. *Static Crosscutting*

Although advice alters the behavior of the system, static crosscutting alters its static structure in a crosscutting manner with structural specification. It is modeled using the metaclass feature. It may be of different types, exception softening, weave-time and warning declaration, or member introduction. A constraint is added to ensure that the static crosscutting stereotype is applied only to classes that are stereotyped Aspect.

IV. CASE STUDY

In order to validate the applicability and efficiency of the proposed profile, we have chosen a simple application that is used frequently in the literature as a motivation example [17]. The Line, Point and FigureElement classes as shown in Figure 2, include the display.update() method as a crosscutting behavior. AspectJ proposes a solution to localize and separate this crosscutting concern by means of an anonymous pointcut and an "after" execution advice as follows:

after(): call(void FigureElement+.set (..))*
|| call(void FigureElement.moveBy(int, int)) {
Display.update();

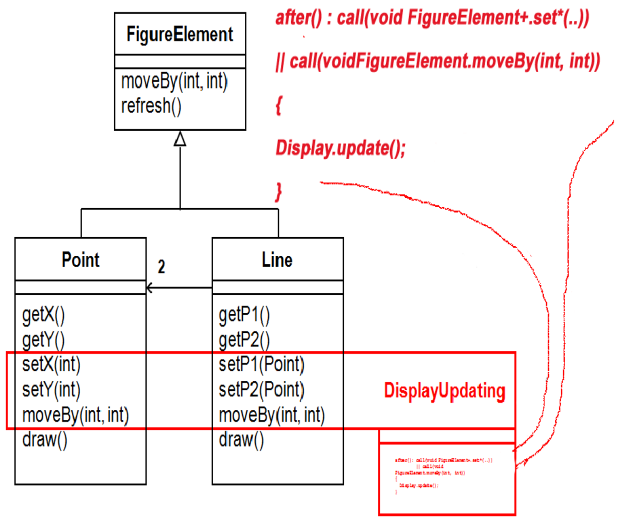


Figure 2. The AspectJ solution for the crosscutting Display.update()method.

In order to use the aspect-oriented paradigm at the design level, we apply our profile to the model. The profile metaclasses became stereotypes and their attributes became tags values with the DisplayUpdating aspect, as shown in Figure 3.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a UML profile as an aspect-oriented modeling contribution based on AspectJ language. Our proposal has several strength points:

- It is a complete specification of the AspectJ language (aspect, advice, pointcut, static crosscutting constructs) in terms of the UML metamodel.
- Compliant with the XMI format, which means that it is possible to manipulate and exchange the profile between UML case tools.

Nevertheless, it remains open to future improvements, namely:

- Generating AspectJ code automatically from the UML model, which is compliant with the XMI standard and fully specified in terms of the metamodel. This could be accomplished by applying MDE/MDA tools and languages, which are already available and mature.
- Demonstrate the applicability and benefits of this profile in various areas. We intend to apply it shortly in the Modeling and Simulation domain.

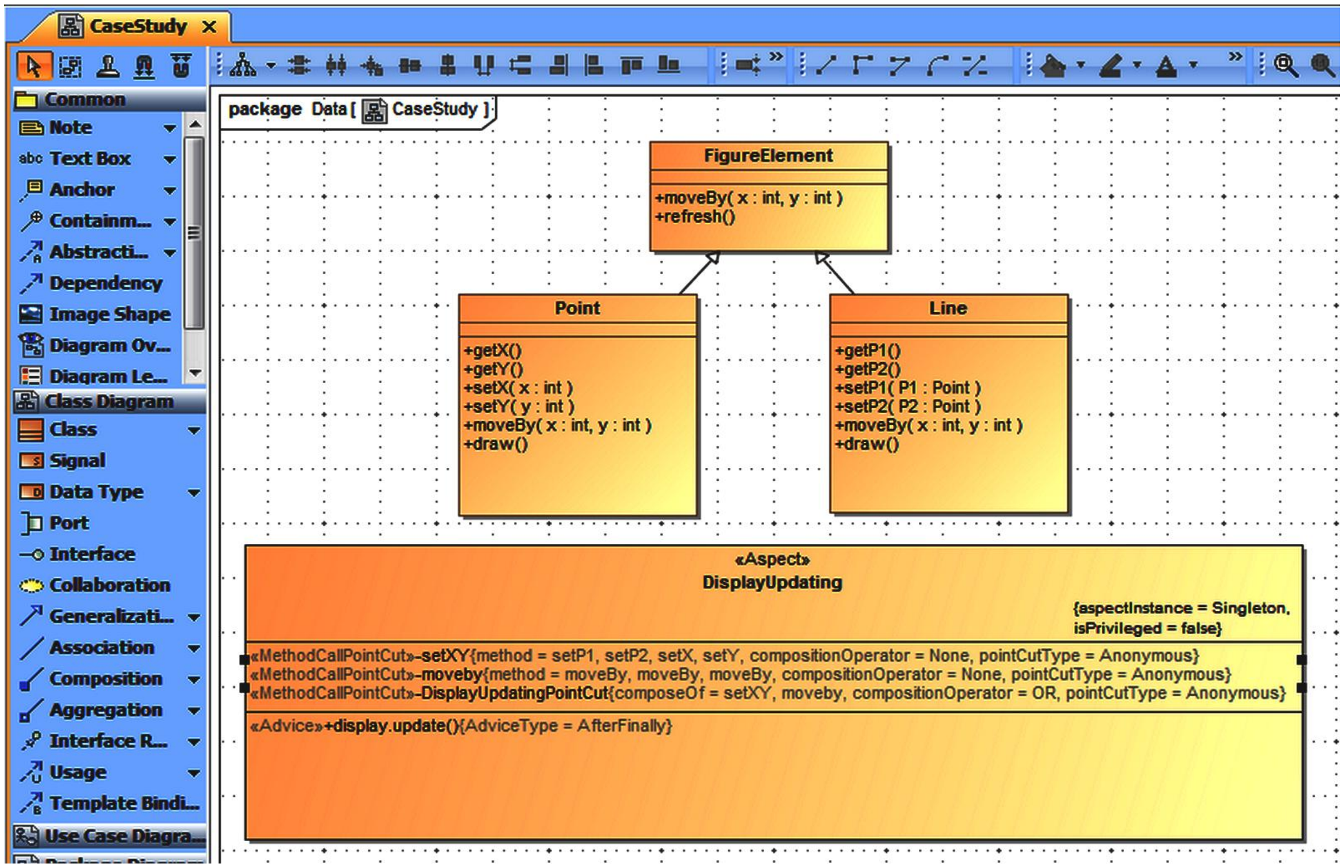

CaseStudy
Common
Note
Text Box
Anchor
Containm...
Abstracti...
Dependency
Image Shape
Diagram Ov...
Diagram Le...
Class Diagram
Class
Signal
Data Type
Port
Interface
Collaboration
Generalizati...
Association
Composition
Aggregation
Interface R...
Usage
Template Bindi...
Use Case Diagra...
package Data [CaseStudy]
FigureElement
+moveBy(x : int, y : int)
+refresh()
Point
+getX()
+getY()
+setX(x : int)
+setY(y : int)
+moveBy(x : int, y : int)
+draw()
Line
+getP1()
+getP2()
+setP1(P1 : Point)
+setP2(P2 : Point)
+moveBy(x : int, y : int)
+draw()
«Aspect»
DisplayUpdating
{aspectInstance = Singleton, isPrivileged = false}
«MethodCallPointCut»-setXY{method = setP1, setP2, setX, setY, compositionOperator = None, pointCutType = Anonymous}
«MethodCallPointCut»-moveby{method = moveBy, moveBy, moveBy, compositionOperator = None, pointCutType = Anonymous}
«MethodCallPointCut»-DisplayUpdatingPointCut{composeOf = setXY, moveby, compositionOperator = OR, pointCutType = Anonymous}
«Advice»+display.update(){AdviceType = AfterFinally}

Figure 3. The UML model after the application of the AspectJ profile.

REFERENCES

[1] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and G. W. Griswold, “An Overview of AspectJ,” Proc. Of ECOOP'01 the 15th European Conference on Object-Oriented Programming, Springer-Verlag London, UK, pp. 327-353.

[2] R. Laddad, “AspectJ in Action”, Enterprise AOP with Spring Applications. Manning Publications, Second Edition, 2009.

[3] M. Forgáč and J. Kollár, “Static and Dynamic Approaches to Weaving,” Proc. Of the 5th Slovakian-Hungarian Joint Symposium on Applied Machine Intelligence and Informatics, Poprad, Slovakia, January 25-26, 2007, pp. 201-210.

[4] Unified Modeling Language (UML), V2.4. http://www.omg.org/spec/UML/2.4/. [retrieved: September, 2013].

[5] I. Groher and T. Baumgarth, “Aspect-Orientation from Design to Code,” Proc. Of the Early Aspects: Aspect-Oriented Requirements Engineering and Architecture Design Workshop In conjunction with 3rd International Conference on Aspect-Oriented Software Development, Lancaster, UK, March 22-26, 2004, pp. 63-68.

[6] O. Aldawud, T. Elrad, and A. Bader, “A UML profile for aspect oriented modeling,” Workshop on Advanced Separation of Concerns in Object-Oriented Systems at OOPSLA2001, 2001, available at http://www.cs.ubc.ca/~kdvolder/Workshops/OOPSLA2001/submissions/26-aldawud.pdf.

[7] O. Aldawud, T. Elrad, and A. Bader, “UML Profile for Aspect-Oriented Software Development,” 3rd International Workshop on Aspect Oriented Modeling at AOSD 2003, Boston, Massachusetts, March 2003.

[8] S. Dominik, S. Hanenberg, and R. Unland, “A UML-based aspect-oriented design notation for AspectJ,” Proc. Of the 1st International Conference on Aspect-Oriented Software Development, AOSD 2002, University of Twente, Enschede, The Netherlands. ACM, April 22-26, 2002, pp. 106-112.

[9] M. A. Basch, “Incorporating Aspects into the Software Development Process in Context of Aspect-Oriented Programming”. UNF Theses and Dissertations, paper 112. University of North Florida, December 2012. Available at: http://digitalcommons.unf.edu/etd/112. [retrieved: September, 2013].

[10] J. Evermann, “A Meta-Level Specification and Profile for AspectJ in UML,” Journal of Object Technology,

Volume 6, no. 7, 2007, pp. 27-49, doi: 10.5381/jot.2007.6.7.a2.

[11] T. Gottardi, R. Aparecida, and V. V. Camargo, "A Process for Aspect-Oriented Platform-Specific Profile Checking," Proc. Of the 2011 international workshop on Early aspects (EA'11), Porto de Galinhas, Brazil, March 21-25, 2011, pp. 1-5, doi: 10.1145/1960502.1960504.

[12] J. U. Júnior, V. V. Camargo, and C. V. Flach, "UML-AOF, A Profile for Modeling Aspect-Oriented Frameworks," Proc. Of the 13th workshop on Aspect-Oriented Modeling (AOM'09), Charlottesville, Virginia, USA, 2009, pp. 1-6, doi: 10.1145/1509297.1509299.

[13] J. Evermann, A. Fiech, and F. E. Alam, "A Platform-Independent UML Profile for Aspect-Oriented Development," Proc. Of the Fourth International Conference on Computer Science and Software Engineering (C3S2E'11), Montreal, Canada, May 16-18, 2011, pp. 25-34.

[14] F. E. Alam, J. Evermann, and A. Fiech, "Modeling for Dynamic Aspect-Oriented Development," Proc. Of the 2nd Canadian Conference on Computer Science and Software Engineering (C3S2E-09), Montreal, Canada, May 19-21, 2009, pp. 109-113.

[15] V. Gowri, "Extending the UML metamodel to grant prop up for crosscutting concerns", International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), Vol. 1, Issue 7, September 2012, pp. 193-198.

[16] MagicDraw Software: http://www.nomagic.com/products/magicdraw.html. [retrieved: September, 2013].

[17] G. Kiczales, E. Hilsdale, J. Hugunin, M. Kersten, J. Palm, and W. G. Griswold, "Getting started with ASPECTJ," Communications of the ACM 44 (10), New York, NY, USA, 2001, pp. 59-65.

A Case Study of Requirements Management: Toward Transparency in Requirements Management Tools

Markus Kelanti, Jarkko Hyysalo, Pasi Kuvaja, Markku Oivo
Department of Information Processing Science
University of Oulu
Oulu, Finland
{markus.kelanti, jarkko.hyysalo, pasi.kuvaja, markku.oivo}@oulu.fi

Antti Välimäki
Metso Automation Inc
Tampere, Finland
antti.valimaki@metso.com

Abstract—**Requirements Management (RM) is a continuous activity that provides an interface between the requirements of engineering and other system development activities. Current literature offers an extensive set of general requirements for RM tools, and several RM tools are available that utilize these requirements. Interviews as a part of a case study to enhance the tool support reveal that the current RM tools do not provide enough transparency to the development process and its activities. The results from these interviews show problems (even with the basic features of RM tools) in decision-making support, reporting, and follow-up of development activities. This paper discusses the problems revealed in the interviews, and suggests further requirements for RM tools to address the problems with transparency.**

Keywords-requirements management; requirements management tools; transparency

I. INTRODUCTION

Requirements Management (RM) is one of the areas perceived as critical in collaborative product development [1], since RM ties together Requirements Engineering (RE) and other product development activities. Therefore, RM has an important role, and it needs adequate tool support for managing the requirements and sharing the information. These tools will ensure the success of product development.

Even though the fundamental activities of RM could be done manually with pen and paper, tools are necessary for practical reasons [2]. The RM tools may offer many features such as a general repository, the ability to import from other tools, communication capabilities, traceability links, change control mechanisms, and information sharing [3]. However, our findings from the interviews with industrial experts show that transparency is not fully taken into account in RM tools. Therefore, we focus the study on identifying transparency requirements that allow the RM tools to provide information about the ongoing status of the development process, enable easy access to relevant information, and make the process more visible and transparent. Thus, our research problem is: *What transparency requirements should be set for RM tools?*

We propose that transparency requirements should be added to the list of requirements for RM tools. Transparency is required in both RM itself and RM tools that will support the developers, help them become aware of the status of development activities and items, and achieve a common, shared understanding about the development goals. All these are necessities in decision making, and help achieve effective and open communication, among other positive impacts, which are all essential for successful, productive development. In short, transparency is the awareness and visibility of what is going on.

The importance of different aspects of transparency and awareness enabling transparency is also recognized in literature [4, 5, 6, 7, 8, 9]. For example, Herbsleb [4] states that if developers have no knowledge what the others are doing, it often leads to misunderstandings in communication content and of motivation. This lack of awareness also makes it difficult to track the effects of changes in distributed collaboration spaces. Transparency in RE in distributed development is especially critical as requirements often emerge from different organizations that challenge the process transparency [9].

Requirements for RM tools already exist in the literature [10, 11, 12]; however, literature about transparency in RM and RM tools is quite scarce. Our contribution focuses on this gap, and we complement the existing knowledge with a new viewpoint—transparency. An industrial case study was conducted in a large global company that develops process automation systems for industrial users. The case study was executed as part of the AMALTHEA project, and it consisted of 11 expert and manager interviews to cover the development process and tools used. The case company uses traditional and agile development methods simultaneously in the same product development project. This kind of setting emphasizes the need for transparency, as the findings of our case study show. The results of the focused interviews with the case company's personnel provided several requirements for transparency-related features and properties for RM tools.

The rest of the paper is organized as follows: Section 2 examines related work; Section 3 outlines the research process; Section 4 presents the empirical study and discusses its results and implications; and Section 5 concludes the study and summarizes the key findings.

II. RELATED WORK

A. RM Tools

RM is a process supporting other RE processes (elicitation, analysis, specification, and verification); it ensures that requirements are documented and traceable, and that changes are properly handled [11, 13, 14]. While requirements form the basis for other development activities, RM provides an interface between RE and the other processes, continuing through the whole product development cycle. Literature defines RM as "the structuring and administration of information from elicitation, derivation, analysis, coordination, versioning, and tracking of requirements during the complete product lifecycle [14]." Several tools are available for managing the RM process [15].

The RM process is generally supported by an RM tool comprised of people assuming roles and responsibilities, processes, and tooling. It also manages the artifacts of the software and systems development process [2]. The tool support should not force specific processes, but should support the developers' tasks and provide the functionalities needed in their work. Current RM tools need to be configured for specific RE and development processes [16].

Literature on the subject provides a comprehensive set of requirements for RM tools and their features [10, 11, 12, 17, 18]. There are also efforts that summarize the available requirements. For example, [14, 17] analyze the literature and classify the RM tool requirements into three categories from the viewpoints of users, project administrators, and IT system administrators. A summary of requirement topics for each category, according to [17], is presented next.

Requirements from the tool users' points of view cover the core functions of an RM tool:

- Information model, views, formatting, multimedia and external files, documentation of history, baselining, traceability, analysis functions, tool integration, import, change management and comments, document generation, collaborative work, checking for offline use, and web access.

Requirements from the project administrators' points of view cover the issues that are not core functionalities but are needed for managing large-scale projects:

- Users' roles and rights, size restrictions, workflow management, and extensibility.

The third category proposes requirements from the tool and the IT system administrators' points of view, which cover the issues related to availability, reliability, and data security:

- Database and encryption.

In addition to dedicated RM tools, most RE tools also support RM; however, their RM capabilities are often inadequate due to a lack of open data model mechanisms, which relate to the recording of user actions, modification of data structures, and standard format of data [18]. Although a wide array of dedicated RM tools is available, and the needs and requirements for RM tools have been recognized in the literature, problems remain with even the basic features of RM tools. For example, requirements for traceability and change management still seem to be difficult issues [19], and both relate strongly to transparency. Most RM tools do not provide adequate support for large distributed projects, nor support the management of large numbers of requests, nor facilitate collaborative RE [16, 20]. There are also usability issues [12, 14] and a lack of support for collaborative work [12].

B. Transparency

Besides these reported problems, we found that the aspects of transparency in RM tools are only partly discussed in the literature. Requirements concerning the awareness of the states of the process and work items are only briefly mentioned under different topics [21, 22]:

- Openness of communication and information sharing;
- Visibility of and access to data, documents, and work items;
- Visibility of decision-making processes and decisions;
- Visibility of processes;
- Transparency of collaboration; and
- Transparency of tools.

Awareness can be defined as the understanding of others' activities, which also provides the context for one's own activities [23]. It is suggested that awareness is the key to transparency [5], and awareness is particularly important in RM [21].

Relevant literature was studied to understand transparency and awareness in an RM context. The following synthesis is based on the literature study and the transparency-related topics that emerged. In the context of RM, transparency can be regarded as the awareness of the following topics:

- *Process support* [11, 12, 14, 17]: It is important to be aware of the states and the histories of software project tasks and the characteristic work activities that describe the environment within which they are performed [24]. Transparent RM tools enable workers to understand the context of their work, which helps them understand their own goals and relate them to others' goals and work. The main concerns are process states, progress, histories, and context.
- *Tooling and work items* [14, 17, 24, 25]: Awareness support is needed to provide information about development artifacts involved in RM in order to have a successful, distributed RM environment [21]. The main concerns are work artifacts, their states and changes, results, documents, data, and context.
- *Decision making* [21, 26, 27]: Awareness about the decision-making process is needed, and forums allow tracking the progress of the states of the requirements. This allows workers to be aware of the person who is working on a particular decision [21]. Forums can also keep track of RE decisions, their rationale, and their effects on software products [28]. The main concerns are decision-making forums, rationale, reasoning process, visibility, and documentation.
- *Collaboration and communication* [6, 22, 24, 25]: RM is often physically distributed work among stakeholders from various organizations [21]. It is important to know what others' roles and responsibilities are, and what they are doing, as it helps to coordinate the collaborative work and diminishes the problem of overlapping work. It is important in RM to understand dependencies, that is, to have the awareness of the other entities that are connected with the one that is being manipulated. This enables an individual to see the impact of one's work on those of others [22]. The main

concerns are visibility of others' actions, skills and competencies, and information access and exchange.

- *Organization and strategy* [6, 29]: Requirements to be implemented need to be synchronized with portfolios and roadmaps that are based on organizational strategy and goals. The RM tools must have the transparency towards organizations' strategies, visions, and goals. For example, Berggren and Bernshteyn [6] suggest "breaking down the strategy into definitive and meaningful components upon which individual employees can act." The main concerns are visions, goals, motives, portfolios, and roadmaps.

The areas of decision making, collaboration and communication, and organization and strategy are often omitted or not addressed extensively in RM tool literature.

III. Research Process and Case Context

The case company uses a project-based approach to develop automation platforms for industrial automation purposes. Interviewees work in the development process, with the aim to improve and implement new functions in those platforms. The development process roughly follows this pattern: requirements elicitation, requirements feasibility analysis, project planning, product design, implementation and testing, and maintenance. In this development process, the purpose of requirements elicitation and feasibility analysis is to gather requirements from different stakeholders, evaluate their technical feasibility and business potential, and generate potential features for an automation platform. One or more features are selected in the project planning phase, where a project is created to implement the selected features. Product design, and implementation and testing are then done for that project. When the feature is released to the customer, it enters the maintenance phase.

A case study was initiated in the company to examine its current RE and RM practices and tools in order to improve them so they would better suit the developers' and managers' needs. The research process used in this case study is shown in Fig. 1.

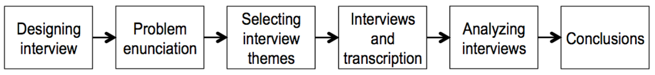


Figure 1. Research Process

At first, relevant research topics were identified with company representatives and researchers, and rough analyses were done on literature, company materials, company presentations, and other sources. Based on this information, the interview was designed to include 12 main questions, each with several sub-questions, to cover current development processes, practices, tools, pros and cons, and possible improvement proposals. A questionnaire template was created and improved in an iterative manner between researchers and the company representative. After a version that satisfied all parties was created, the actual interviews were conducted. The final questionnaire comprised 11 main questions covering the following topics: terminology, currently used methods and processes, tools, information needs and uses, responsibilities, and pros and cons perceived by the interviewees.

In total, 11 interviews were planned and performed in the case company. Seven of the interviewees were designers and engineers working in the development process, and four were managers from different levels in the organization.

The interviews were executed over a period of nine months. Seven interviews were conducted during the autumn of 2012, and four during the winter of 2012-2013. The duration of each interview was approximately 1.5 hours. The questionnaires were delivered to the interviewees in advance, so they could prepare themselves for the interview. Two researchers conducted the interviews, mostly face-to-face. All of the interviews were recorded and transcribed, and the researchers wrote a short summary of each transcription. The summaries and transcriptions were sent to the respective interviewees within two weeks, and they were given one week to validate the information. The feedback and change requests were taken into account during analysis, but only a few interviewees made any (minor) corrections or added anything to the summaries.

In the next step, the validated information was analyzed to find themes in the content. Nvivo 10 was used to store the interview data and to help facilitate the analysis process. Nvivo 10 was selected mainly due to the researchers' familiarity with the tool, its support for the different coding techniques applied for the data analysis, and theme identification. The interview data was auto-coded first based on the questions on the interview template. The next step was to analyze and code the data to find major themes from the interviews. The interviewees also reviewed these analyses individually, and a workshop was arranged with them to discuss the results further. Based on the interviews, analyses, reviews, and the workshop, we identified one major theme—transparency in RM tools. After this, the data was analyzed to find the transparency requirements. The researchers also analyzed and coded the data to identify other possible themes related to transparency, based on the literature review.

IV. Findings and Analysis

The case company uses three main tools to manage its requirements: Jira, Polarion, and a proprietary application developed in-house. The case company uses different development methods, depending on the system under development and the technology involved. Jira is mainly employed to manage agile projects, while Polarion is used only for safety-critical systems. The proprietary application is used to store information about requirements and features, and it supports other development methods applied in the case company (ranging from adapted waterfall methods to agile approaches). It offers basic information fields and the functionality to record, link, and store the data into a database accessible by users. The proprietary tool is currently used to manage all requirements and features. Other tools used in the process are mainly Microsoft Office products like Word, Excel, Visio, and PowerPoint, as well as some tools developed in-house for testing and demonstration purposes.

Based on the interviews, it became obvious that the current tools used in RE are unable to provide visibility, easy access to information, or knowledge about what is happening in the development process at any given time. Throughout the interviews, the respondents constantly raised the issue of their inability to access information relevant to their work. This problem causes unnecessary resource consumption in the decision-making, reporting, and follow-up areas of the development activities. These issues concentrate on the RM aspect of these tools, and after comparing them with the literature about RM tools [12, 14, 17, 21], we concluded that there is a gap within the transparency aspect of these particular tools used in the case company's RE process.

The following sections present the identified transparency requirements and the requirements that affect transparency, with references to existing literature, if the latter backs up the requirements. The sections are divided according to the transparency categorization presented in the related work section. The identified requirements are summarized in Table I.

A. Process Support

R1) RM tool shall provide information about the states of the process and tasks

Literature suggests that an integral part of the development process is the awareness of the states and histories of software project tasks and work activities [24].

Understanding the current state is needed to enable the developers to react to changes and unexpected events. It also builds a shared understanding, which is an integral part of cooperative development [24].

Both managers and engineers voiced the need to know the status of a task or a process. The most commonly mentioned situation for this is when customers request information about the development status, especially estimations for when a product will be ready for delivery or piloting. Currently, this information is not easily available, and sometimes months can pass before any information reaches the customer. On the other hand, management may need information about the status of a project to make estimations and check whether the schedule and resources are up to date. This information needs to be collected manually, since current tools are inadequate.

R2) RM tool shall only show the task-relevant information

Interviewees commented that some of the tools they use tend to display a lot of information: status, historical data, design documents, comments, and so on. This helps improve transparency, but if the information is irrelevant to the current task, it overwhelms the users. Another danger is when the information is not updated regularly in the RM tool, but kept in separate documents in other databases, on developers' PCs, or in emails. This problem was also revealed by interviewees. Therefore, to support transparency, task-relevant information must be available and easily accessible, without any additional effort.

R3) RM tool shall support the actual development tasks

Related to the relevant information for tasks, the RM tool should obviously support the actual development tasks. Some of the interviewees are more engaged with agile development methods, and they commented that both Jira and Polarion are more suitable for their work. According to them, both tools are better designed for the development tasks used in either agile processes or safety-critical applications. Only necessary information for a development task should be visible in the tools used by the developers.

R4) RM tool shall provide task guidance

Heuristic knowledge and providing ways-of-working to guide developers while performing systems development are needed. They are useful, for example, for decision-making purposes or activities needed to create the conceptual specifications of the system [30]. A transparent RM tool should not only help workers understand the work context and its goals, and relate them to others' goals and work, but also provide guidance about what kinds of information workers need to produce in the development tasks.

For example, the developers reported in the interviews that financial estimations are especially essential in several tasks, but it is very hard to estimate with the current tools and available information. These estimations are used in different parts of the process to make decisions, and it is important to know how to do those estimations, and in what format the information should be documented.

TABLE I. TRANSPARENCY REQUIREMENTS FOR RM TOOLS

Topic	#	Requirement for RM tool	Related work
Process support	R1	Provide information about the state of the process and tasks	[14, 17, 24]
Process support	R2	Show only the task-relevant information	
Process support	R3	Have task views that match the actual development tasks	[14, 17]
Process support	R4	Provide task guidance	
Process support	R5	Provide process guidance	
Tooling and work items	R6	Provide information about development artifacts	[24]
Tooling and work items	R7	Provide standard information templates for RE items	
Tooling and work items	R8	Support linking	[17, 23, 24]
Tooling and work items	R9	Maintain link validity	
Tooling and work items	R10	Enforce linking rules among items	
Tooling and work items	R11	Support traceability	[1, 11, 12, 14]
Tooling and work items	R12	Support version control	[11, 12, 14, 17, 24, 25]
Decision making[a]	R13	Provide the rationale and reasoning process for decisions	[24]
Decision making[a]	R14	Provide visibility of decisions and their documentation	[28, 29]
Decision making	R15	Be able to generate status reports from processes	[12, 14, 17, 21, 24, 25]
Collaboration and communication[a]	R16	Provide awareness of others' actions	[24, 25]
Collaboration and communication	R17	Provide support for information sharing between management and developers	
Collaboration and communication	R18	Enforce a coherent terminology for RE items	[24, 29]
Organization and strategy	R19	Support breaking down the strategy, vision, goals, and motives into work tasks	[6, 24, 25]
Organization and strategy	R20	Provide information about available resources, skills, and competencies	[21, 26, 27, 28]

[a] These requirements are suggested by the literature, but not specifically mentioned in the interviews.

R5) RM tool shall provide process guidance

Furthermore, when this information is produced, it is not clear where and how it is utilized. Management needs information about the current state, and to get it in the form they need, it has to be inserted in a certain format and from a certain viewpoint. If users are not presented with proper guidance to create information, including where and how it will be used, it will not be as reliable as it should be. This is especially true in RM tools, where accurate information is crucial. While these requirements do not directly support transparency itself, without them, the information will not serve its purpose and can even cause negative outcomes.

B. Tooling & Work Items

R6) RM tool shall provide information about development artifacts

In software development, the artifacts are mostly documents and code. Literature suggests that awareness support is needed to provide information about development artifacts involved in RM in order to have a successful, distributed RM environment [21]. This awareness provides up-to-date information to stakeholders for better decision making.

Generally, the results from the interviews indicate a clear need to access information regarding any item in the development. These items include a single requirement, project status, use case, original request, and so on. The main reason is quite clear—interviewees need more information in order to perform their tasks. They often also need old documentation, previous work items, or other items linked to the item they are working on. This is true for both engineers and managers, and both commented that it is important to access information about a single item in order to learn its status, who is working on it, and generally understand its status.

R7) RM tool shall provide standard information templates for RE items

In the process, standard templates are used for documenting needs and requirements that contain basic information necessary for determining business potential, technical feasibility, and other relevant information for decision making. This is considered a good practice in general, but interviewees pointed out that these templates need to match the information needs of the tasks or activities at hand.

R8) RM tool shall support linking

The literature also discusses how one's work may impact those of others [22]. This includes artifacts and associated tasks, collaborators, and the concurrent work context of collaborators and resources [24]. Awareness of the context and others' actions makes it possible for developers to structure their interactions and cooperative processes, and to provide a context for one's own activities [23, 24].

Interviewees generally agreed that one of the main functions in the tools they use is the ability to link different items. This functionality is considered necessary to show dependencies and relationships among different requirements, features, and products. The ability to link different items is essential to the developers, particularly, how changes they introduce will affect different parts of the platform they develop. Since many developers work on a single platform or product, it is important to know the relevant items others are working on.

R9) RM tool shall maintain link validity

Another challenge related to linking different items in a tool is that the links sometimes connect to the wrong versions of the development artifacts. This can cause wrong versions to be implemented and tested. It also becomes increasingly difficult to search for information. This is especially true when data are searched after some time, and the item is not in the fresh memory. Developers clearly need to access valid information that points to the correct, updated version. If the validity is ignored, the link itself becomes useless. If this functionality is ignored, it can lead to situations where wrong versions are used in the work, and conflicts will arise.

R10) RM tool shall enforce linking rules among items

However, just enabling functionality to link and keep the links up to date is not enough. Interviewees also commented that linking practices should be enforced to keep the links coherent and understandable. Current tools in the case company allow anything to be linked in several different ways, with no generally accepted conventions for their use. This has led to unnecessary complexity with the database and tool, as individuals follow their own preferences. It was suggested that there should be rules and restrictions on the kinds of links to be used and the ways they should be described. The RM tool should enforce these rules to maintain cohesion, which will enable better transparency.

R11) RM tool shall support traceability

Traceability is one of the basic functions and requirements for RM tools. Traceability is needed to maintain and follow the relationships among requirements and design, implementation, and test artifacts [10]. With good tool support, traceability could enable analysis that would otherwise require more effort [17].

This is also one of the key functionalities, according to interviewees. On several occasions, interviewees mentioned that lack of traceability is troublesome because it hides what has already been done for a requirement. When this happens, they have to investigate what has been done in order to understand how the item has been developed in the past and where it originated. Testing would benefit if they could trace the requirements back to their original sources to see how things should work in the system.

R12) RM tool shall support version control

Enabling traceability has also led to a demand for proper version control, since this is lacking in most of the current tools. Without version control, it would be hard to know what has been done for any given item in the process.

C. Decision-making

R13) RM tool shall provide the rationale and reasoning process for decisions

To support decision making, the RM tool should provide identified criteria for evaluating the achievements. Moreover, decisions need to be explained and transparent for all relevant stakeholders. This improves the overall effectiveness of the RE process and provides understanding about the nature of the decisions made. It is necessary to keep track of RE decisions, including their rationale and effects on the product [28].

For managers and developers, the decisions are made mostly among the relevant parties, and the rationale is generally available for the interested stakeholders. Even in this case, knowing the rationale for a decision is still important, and the interviewees mentioned the times when they might need to communicate results to a customer.

R14) RM tool shall provide visibility of decisions and their documentation

The literature suggests ways to provide visibility of decisions. Decisions need to be documented and fed back into the system, so the workers can benefit from the experience [21]. Decisions also need to be integrated into organizational information systems; this allows them to be better understood by relevant stakeholders [26].

Customers often present their needs and wait for the company to react to them; all of the interviewees pointed out that customers should be told the reasoning for the decision when it finally comes. Interviewees expressed that this information should be available in the RM tool, either directly visible for the customer or for the developers to inform the customer.

R15) RM tool shall be able to generate status reports from process

One of the main concerns for managers is that the current company tools do not allow them to generate status reports such as project status, feature status, portfolio, and overall status reports from several projects. They commented that they can access some of the necessary information in the existing tools, but the tools should only provide the information they need and not just everything that is available. Due to the lack of this kind of functionality, the management has to collect the information by asking each project manager individually in order to generate the reports themselves.

Managers also expressed a clear need for constant reporting support from the tool. They especially need up-to-date reports on the various projects they are managing in order to track problems, delays, and progress in general. Project managers need to communicate information to upper management and customers about the schedule and progress. Portfolio reports, project reports, or feature status reports were all mentioned as important. The RM tool would therefore need to synthesize reports on the basis of need.

"I think that this kind of upper-level project management is not possible with the current tool. And this kind of overview to all projects is missing. One has to pick up the pieces of information to create the overview. That is the biggest shortcoming in the tool, in my opinion." (Interviewee)

D. Collaboration & Communication

R16) RM tool shall provide awareness of others' actions

The RE is inherently distributed [21]; thus, there are awareness needs in RE and RM. In collaborative work, it is important to know what others' roles and responsibilities are, and what they are doing, because it helps diminish the problem of overlapping work. It is also highly relevant to have knowledge of others' interactions with the space and its artifacts. This helps with understanding who is working with what artifact and the artifacts of interest [7, 24].

While it is not necessary to know what a single developer or manager is doing at a certain moment, interviewees mentioned the need to generally know what is happening. This information is considered useful for making plans for future projects and for usage of resources, from management's perspective.

R17) RM tool shall provide support for information sharing between management and developers

Interviewees also said that transparency among different units, developers, and management would result in better understanding about the business and the real-world use potential of the products. This is not only tied to RM tools; often they are the tools used by management, while developers are the most important source of information in this area. Therefore, to establish proper transparency through information sharing, the RM tool needs to enable information flow from developers to management.

R18) RM tool shall enforce a coherent terminology for RE items

The relevance of information changes across different contexts; thus, the context should always be understood. As previously mentioned, understanding their work context enables workers to understand their own goals and relate them to others' goals as well. For example, Basili *et al.* (2007) suggest that "context specification is an important part of defining goals and deriving measures, since it prevents drawing wrong conclusions from the analysis" [29]. The evolving internal and external state of information characterizes the situation of entities in a shared environment [24].

During the interviews, the understanding of RE concepts (such as requirements, features, RE, and RM) varied from one interviewee to another; they often had different terms for similar concepts. Between the engineers and managers, this does not cause too much trouble because they are able to communicate face-to-face, but when they communicate with someone in another location, these differences are a potential source of misunderstanding.

E. Organization & Strategy

R19) RM tool shall support breaking down the strategy, vision, goals, and motives into work tasks

Transparent goals help the collaborative work and improve efficiency by reducing redundant work. Strategy transparency can be stated as "breaking down the strategy into definitive and meaningful components upon which individual employees can act" [6]. Strategies, visions, goals, and motives should be transparent and understood at all levels of work, and defining the portfolios and roadmaps based on organizational strategy and goals is suggested.

Interviewees expressed a need to see the plans and short-term roadmap for any automation platform they develop. They commented that it helps them decide what is needed and what areas they should prioritize. If this functionality would be available in the RM tool itself, it would remove the need to use time and other tools to find the information they need in their work.

R20) RM tool shall provide information about available resources, skills, and competencies

An integral part of the process is the awareness of the expertise of the developers working on the project [24]. A clear understanding about the availability of the talent pool in the organization enables the alignment of talents with the organizational strategy and development tasks.

During the interviews, both managers and developers expressed the need to access information regarding the available resources and competencies within the company. Managers need better information about the resources available for project planning, so they can satisfy the customers' needs and schedule the releases. Developers need to know about persons who can provide further information or clarification for requirements, in case the existing information is not sufficient.

V. Conclusion and Future Work

In this paper, we studied RM and RM tools in a large global organization that develops process automation systems. In a collaborative setting, different organizations, or even teams within an organization, may use various development methodologies and tools, causing challenges for RM; thus, support for transparency is required.

Based on our findings from the interviews and literature, RM tools should support transparency and provide the features needed for awareness creation. This paper has presented a set of necessary requirements for RM tools to support transparency. We have categorized these requirements under the following topics: process support, tooling and work items, decision making, collaboration and communication, and organization and strategy. We have also emphasized those transparency requirements that are already included in the requirements list for RM tools, but are still regarded as inadequately addressed.

A. Case Validity and Limitations

Study validity was addressed in several ways. Construct validity was dealt with through an extensive literature review, comparison of previous findings with current research using multiple sources of evidence, and utilization of key sources as reviewers. Internal validity regarding cause-effect relations was handled via multiple sources of evidence and iterative research, which gradually built the outcome. External validity involving the generalization of the results was tackled by having different organizational units as evaluation platforms. While the interviews were conducted only in one company in the automation domain, the literature supports the findings in different domains. However, a study in other organizations may introduce new requirements for transparency. The purpose of this study is not to suggest statistical generalizations but to enable generalization of the results to cases that have common characteristics. For further generalization, more studies are required. Finally, reliability was managed with rigorous research protocol, documentation, data collection procedures, and peer reviews.

B. Implications for Research and Practice

These results should interest both researchers and practitioners, since transparency requirements for RM tools are not extensively discussed in the literature. This study provides insights for academic research and lays the groundwork for further scholarly inquiry, for example, in validating the results in other domains and development phases.

Practitioners could learn to understand the importance of transparency in RM and RM tools, and thus have those requirements implemented in the tools. If transparency is addressed adequately, it can also benefit the practitioners by enabling better decision making and information flow in the development processes. Transparency will also help the development process and improve product quality, as well as the efficiency of the development.

C. Areas for Future Work

There is still a place for further work, and our intention is to validate the findings in the telecommunication and automotive industries. We also aim to have transparency requirements taken into account in applications other than RM tools. Additionally, RM tools should still be able to monitor and provide support for users, even if different development methods are used to build the systems. The needs of different development methods are another area for future work. Finally, we intend to implement the requirements in a prototype tool for practical validation and evaluation purposes in a follow-up study, where we will also examine how currently available RM tools conform to the transparency requirements presented in this paper.

Acknowledgment

This research is supported by the European ITEA2 program with national funding from Tekes (the Finnish Funding Agency for Technology and Innovation). The authors would like to thank AMALTHEA project partners for their assistance and cooperation.

References

[1] J. Hyysalo, P. Parviainen, and M. Tihinen, "Collaborative Embedded Systems Development: Survey of State of the Practice," Proc. IEEE Symp. Engineering of Computer-Based Systems, (IEEE 13), 2006, pp. 130–138.

[2] O. Gotel and P. Mäder, "Acquiring tool support for traceability," in Software and Systems Traceability. Springer London, 2012, pp. 43–68.

[3] K. E. Wiegers, "Automating requirements management," in Software Development, 7.7, 1999, pp. 1–5.

[4] J. D. Herbsleb, "Global software engineering: The future of socio-technical coordination," in Future of Software Engineering, IEEE Computer Society, 2007, pp. 188–198.

[5] M. Beaudouin-Lafon and A. Karsenty, "Transparency and Awareness in a Real-time Groupware System," Proc. ACM Symposium on User Interface Software and Technology, (ACM 05), 1992, pp. 171–181.

[6] E. Berggren and R. Bernshteyn, "Organizational Transparency Drives Company Performance," Journal of Management Development, 26.5, 2007, pp. 411–417.

[7] C. Gutwin, S. Greenberg, and M. Roseman, "Workspace Awareness in Real-time Distributed Groupware: Framework, Widgets, and Evaluation," Proc. International Conference on Human–Computer Interaction: People and Computers (11), Springer London, 1996, pp. 281–298.

[8] R. E. Grinter, "Using a Configuration Management Tool to Coordinate Software Development," Proc. Conference on Organizational Computing Systems, (ACM), 1995, pp. 168–177.

[9] D. C. Gumm, "Distribution dimensions in software development projects: a taxonomy," in Software, 23.5, IEEE, 2006, pp. 45–51.

[10] I. Sommerville and P. Sawyer, Requirements Engineering: A Good Practice Guide. John Wiley & Sons, 1997

[11] G. Kotonya and I. Sommerville, Requirements Engineering: Process and Techniques. John Wiley & Sons, 1998

[12] M. Lang and J. Duggan, "A tool to support collaborative software requirements management," in Requirements Engineering, vol. 6, no. 3, 2001, pp. 161–172.

[13] K. E. Wiegers, Software Requirements. Microsoft Press, 2000.

[14] M. Hoffmann, N. Kuhn, M. Weber, and M. Bittner, "Requirements for Requirements Management Tools," Proc. IEEE International Requirements Engineering Conference, (IEEE 12), 2004, pp. 301–308.

[15] INCOSE The International Council on Systems Engineering, "Tools survey: RM tools," http://www.incose.org/ProductsPubs/products/rmsurvey.aspx (referenced February 8, 2013.)

[16] A. Finkelstein and W. Emmerich, "The Future of Requirements Management Tools," Information Systems in Public Administration and Law. Austrian Computer Society, 2000.

[17] D. Beuche, A. Birk, H. Dreier, A. Fleischmann, H. Galle, G. Heller, D. Janzen, I. John, R.T. Kolagari, T. von der Maßen, and A. Wolfram, "Using Requirements Management Tools in Software Product Line Engineering: The State of the Practice," Proc. Conference on Software Product Line Engineering, (IEEE 11), 2007, pp. 84–96.

[18] J. M. Carrillo de Gea, J. Nicolás, J. L. Fernández Alemán, A. Toval, C. Ebert, and A. Vizcaíno, "Requirements Engineering Tools: Capabilities, Survey and Assessment," Information and Software Technology, vol. 54, no. 10, 2012, pp. 1142–1157.

[19] G. Sabaliauskaite, A. Loconsole, E. Engström, M. Unterkalmsteiner, B. Regnell, P. Runeson, T. Gorschek, and R. Feldt, "Challenges in aligning requirements engineering and verification in a large-scale industrial context," in Requirements Engineering: Foundation for Software Quality, Springer Berlin Heidelberg, 2010, pp. 128–142.

[20] P. Laurent and J. Cleland-Huang, "Lessons learned from open source projects for facilitating online requirements processes," in Requirements Engineering: Foundations for Software Quality, Springer Berlin Heidelberg, 2009, pp. 240–255.

[21] D. Damian, J. Chisan, P. Allen, and B. Corrie, "Awareness Meets Requirements Management: Awareness Needs in Global Software Development," Proc. International Workshop on Global Software Development, International Conference on Software Engineering, 2003, pp. 7–11.

[22] M. A-D. Storey, D. Cubranic, and D.M. German, "On the Use of Visualization to Support Awareness of Human Activities in Software Development: A Survey and a Framework," Proc. ACM Symposium on Software Visualization, (ACM), 2005, pp. 193–202.

[23] P. Dourish and V. Bellotti, "Awareness and Coordination in Shared Workspaces," Proc. ACM Conference on Computer Supported Cooperative Work, (ACM), 1992, pp. 107–114.

[24] I. Omoronyia, J. Ferguson, M. Roper, and M. Wood, "A review of awareness in distributed collaborative software engineering," Software: Practice and Experience 40.12, 2010, pp. 1107–1133.

[25] W. Prinz, H. Löh, M. Pallot, H. Schaffers, A. Skarmeta, and S. Decker, "ECOSPACE – Towards an Integrated Collaboration Space for eProfessionals," Collaborative Computing: Networking, Applications and Worksharing, IEEE, 2006, pp. 1–7.

[26] G. Ruhe, "Software engineering decision support–a new paradigm for learning software organizations," Advances in Learning Software Organizations, vol. 2640, Springer Berlin Heidelberg, 2003, pp. 104–113.

[27] C. Jensen and W. Scacchi, "Collaboration, Leadership, Control, and Conflict Negotiation and the netbeans.org Open Source Software Development Community," Proc. Hawaii International Conference on Systems Sciences, (IEEE 38), 2005, p. 196b.

[28] A. Aurum and C. Wohlin, "The fundamental nature of requirements engineering activities as a decision-making process," Information and Software Technology 45.14, 2003, pp. 945–954.

[29] V. Basili, M. Lindvall, M. Regardie, C. Seaman, J. Heidrich, J. Münch, D. Rombach, and A. Trendowicz, "Bridging the gap between business strategy and software development," Proc. International Conference on Information Systems, (ICIS 28), 2007, pp. 1–16.

[30] C. Rolland, C. Souveyet, and M. Moreno, "An approach for defining ways-of-working," Information Systems 20.4, 1995, pp. 337–359.

Two-Hemisphere Model Based Approach to Modelling of Object Interaction

Oksana Nikiforova, Ludmila Kozacenko, and Dace Ahilcenoka
Faculty of Computer Science and Information Technology
Riga Technical University
Riga, Latvia
oksana.nikiforova@rtu.lv,ludmila.kozacenko @rtu.lv, dace.ahilcenoka@rtu.lv

***Abstract* — It is a modern trend to use automatic transformations of different type of models to develop a software system. Software engineers have quite enough notations to present models at different levels of abstraction and at different stages of software development project. UML is an industrial standard for system modeling and specification and offers notational conventions for presentation of both aspects of the system – dynamic as well as static one. Currently, the research focuses in the area of software system modeling and model transformation is turned exactly to the dynamic aspect of the system. We propose to use the so called two-hemisphere model for receiving a set of elements, which are used for modeling an object interaction as a central part of the system dynamic presentation. The paper describes the main principles of the two-hemisphere model transformation into the UML sequence diagrams, as well as compares it to other transformation approaches.**

Keywords - UML sequence diagram; two-hemisphere model; layouting algorithm; model transformation; BrainTool.

I. INTRODUCTION

As we had stated in a previous paper devoted to the two-hemisphere model-driven approach [1], tools to support models and modeling at the initial stage of software development is the modern trend in business process modeling and analysis. Therefore, the focus of the automation of software development is shifted from automatic code generation from the system model to the automatic modeling of the problem domain and further code generation from them. Here, the valuable notation became the Unified Modeling Language (UML) [2] and its class diagram, which specifies the structure of the developed system and static information about system behavior. An ability to generate elements of the UML class diagram from the two-hemisphere model by BrainTool is demonstrated in [1]. Currently, we consider the dynamic aspect of the system and are investigating an ability to generate elements to present an object interaction according to the UML notation [2].

In general, there are two ways of looking at any software system. One way is to consider just data, including variables, arguments, data structures and files where the operations are examined only within the framework of the data. And the other way of viewing the software system is to consider just the operations performed on the data where data are of the secondary importance. According to the object-oriented software development, data and operations are viewed at equal importance, in spite of the fact that sometimes data have to be stressed and other times operations are more critical.

The main attention during object-oriented software development is devoted to the definition of system objects which are the primary artifacts of the developed system and include the information about data and operations together. Therefore, one of the fundamental tasks during object-oriented software development is to define an object structure and to share the responsibilities of an object, i.e., to determine the operations for objects to perform.

The paper presents the way to solve the problem of sharing responsibilities between objects by using the two-hemisphere model supported by BrainTool. We illustrate the process creating a two-hemisphere model [4] for a business domain and then investigate construction of the UML sequence and communication diagrams. In order to solve this task we defined a set of transformation rules and also focused on the problem of automatic layout of the UML diagrams after their derivation from the two-hemisphere model. Since it is very important to ensure that the diagrams are well built not only in terms of their content, but also how they visually represent the information.

The paper is structured as follows. Section 2 explains the essence of the object-oriented software development and discusses the importance of the object definition and responsibilities shared between them during the object oriented system analysis and modeling. Section 3 defines the essence of the two-hemisphere model transformation to share responsibilities to perform operations by system objects. Section 4 demonstrates BrainTool supporting the proposed approach and discusses the problem of the UML sequence diagram layout and its solution. Section 5 compares BrainTool with other tools giving an ability to create the UML diagrams. In conclusion, we stress the main contribution of the paper and state the directions for the further research.

II. THE ROLE OF THE OBJECT INTERACTION MODELING IN THE SOFTWARE DEVELOPMENT

The object-oriented software development assumes that the main attention is to be devoted to identification of objects from the problem domain and to sharing responsibilities of operation execution between these objects. Therefore, the role of the system modeling becomes very important. In the object-oriented software development, the standard notation for the system modeling is the Unified Modeling Language

(UML) [2]. The UML diagrams give a possibility to present different aspects of software system, but UML is just a notation and does not contain methodological instructions on how to model the system. The developer needs the information about the system to be developed in the form, which gives an ability to transform this information into UML diagrams.

Basically, the software system development starts with the business information gathering and presenting it in the form suitable for further software system modeling. In classical approach, this information is presented as the processes to be performed and the information flows required for the process execution. Then, this presentation of business information has to be transformed into the model, which in object-oriented manner for software development requires to present objects to interact in the form of UML sequence diagram [2]. It shows the objects, their lifelines and messages to be sent by objects-senders and performed by object-receivers and is used to present dynamic aspect of the system, which in object-oriented approach is expressed in terms of message sending among objects. The dynamic of interactions is defined by an ordering of the messages. It serves as a basis for definition of operations performed by objects to be grouped into classes, as well as to present and to verify a dynamic aspect of class state transition. The problem, which is recently widely researched in the area of the object interaction analysis, is formal transitions among the models presented at the level of problem domain and system presentation expressed in terms of the object interaction, if we are dealing with the object-oriented software development and using a set of model transformation rules. For now, this transition is defined and is partly supported by UML modeling tools and some guidelines exist on how these transformations should be performed.

Loniewski et al. [3] show the result of analysis of different approaches to transformation of the problem domain description into the UML class diagram during the last 10 years, published in four digital libraries (IEEEXplore, ACM, Science Direct, Springerlink). The survey states that there exist enough approaches with different types of solutions for the generation of a UML class diagram and half of them are automated and supported by tools. However, the authors of [3] stress that these tools are not widely used practically and are created to approve the automation level of the approach offered by their vendors. Other researchers who are investigating the functionality of the UML modelling tools and model transformation tools raise the question about the ability to define the tool chain to cover all the necessary activities for software system development. For example, the lack of a conceptual view on the integration problem and appropriate reuse mechanisms for already existing integration knowledge, which forces the developer to define model transformation code again and again for certain recurring integration problems in an implementation-oriented manner, resulting in low productivity and maintainability of integration solutions. We consider that the maturity level of advanced modelling and model transformation tools is not enough to support the full chain of software system development. Thereby, despite the number of approaches to automatic creation of the system model and further code generation from it, the variety of tools supporting the system modelling at the initial stage of software development are reduced to UML editors and "tight" code generators.

The core of this paper is a hypothesis that our proposed notation of the two-hemisphere model supported by BrainTool contains enough information for sharing responsibilities among objects and can serve for automatic generation of the elements to present the UML sequence diagram. Whereas UML sequence is stated as an one of ambiguous UML diagrams [5], with the implicit and informal semantic that designers can give to basic sequence diagram as a result of this conflict [6], [7], [8]. The two-hemisphere model [4] contains information about business processes and concepts and has already been used for representation of object interaction with UML communication diagram [9], where only static view of the system is investigated and an ordering of message sending and receiving is missed. Currently, we define the mapping between elements of two-hemisphere model and elements of UML sequence diagram, especially in its timing aspect, solve the problem of sequence diagram layout and offer to use BrainTool for receiving of the UML sequence diagram.

III. Definition of transformation from the two-hemisphere model into the UML sequence diagram

A nature of transition from business information into the object interaction is found in the definition of which processes have to be performed in the system and which performer will execute exact process at the software level of system modeling. In order to identify a performer of the process at the software level of system presentation the process has to be analyzed with the aim to define a software operation to execute the process and to notice the object to perform this operation. So far two general steps can be defined for the object-oriented system analysis. The first one is to identify objects themselves. This task is solved by [10], [11]. In general, the analysis of entity relationship [12] can serve as a base for the object identification of the software system. Further, the second activity of object-oriented system analysis is so called "sharing of responsibilities" among the objects, which is not so trivial and is stated for solving by the author of the paper. The main task to be defined is which operation will be executed by which object and in which time sequence.

In UML models, objects interact to implement behavior. UML has two kinds of diagrams to reflect object interaction – communication and sequence diagrams. Communication diagram allows observing the common interaction of objects in the system mainly focused on associations between objects and time aspect is not stressed in the communication diagram. The UML sequence diagram shows interaction of objects for execution of concrete use case or business function expressing time aspect as a main focus of the modeling. We analyze the possibility to generate all the necessary information for object interaction (especially time component of that) in terms of the UML sequence diagram.

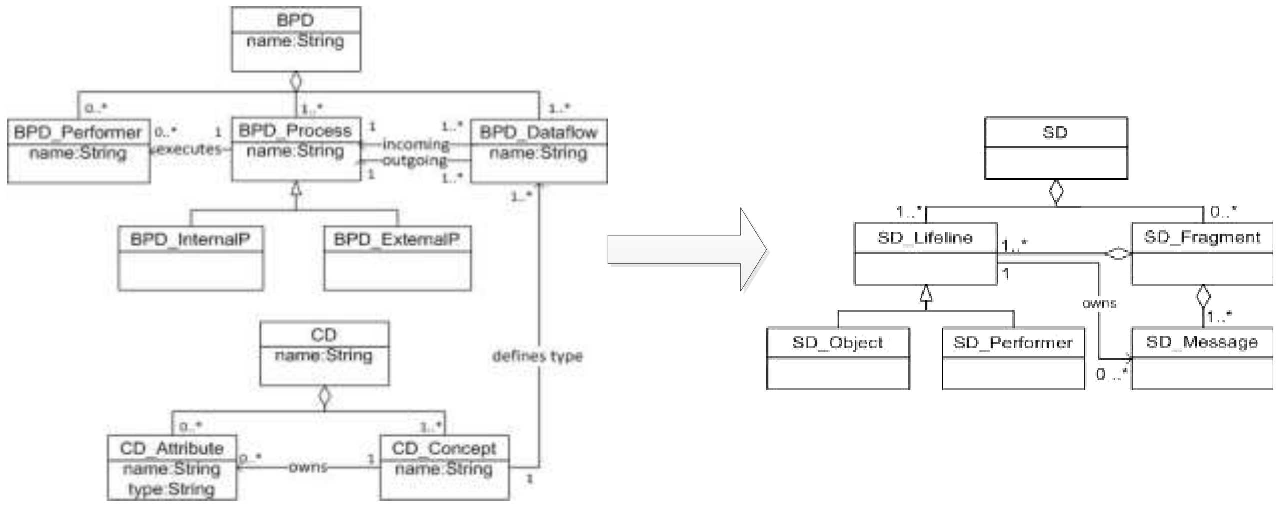


Figure 1. Elements of the two-hemisphere model used in transformation rules to generate elements of the UML sequence diagram.

The definition of elements of the sequence diagram needs an examination of elements of two-hemisphere model, which is presented as a business process model related with concept model. The sequence diagram consists of objects, their lifelines and messages which they have to send to other objects.

A simplified sequence diagram metamodel [13] presented at the right side of Fig. 1 shows only those elements of the diagram and their dependences, which are being used in the transformation process, in other words, only those sequence diagram elements, which can be acquired from a two-hemisphere model. The left side of Fig. 1 shows the metamodel of the two-hemisphere model [13].

Object identification is based on the analysis of noun phrases in the problem domain description [10], where it is presented in the form of two-hemisphere model and contains the information about the problem domain, where the noun phrases are defined for the events (arcs) of business process model and concepts of the concept model (see Fig. 2). Therefore, it is possible to suggest that description of an event in business process with its defined data structure in concept model can serve as a basis for identification of an object in the sequence diagram.

The transformation of the two-hemisphere model into the UML communication diagram is performed in a direct way of graph transformation, where arcs (i.e., information flows in Fig. 2) of graph of business processes are transformed into the nodes of graph of object communication. E.g., “Applicant data” as an information flow in process model becomes a class “Applicant data” in communication and sequence diagrams. Process “add applicant to group” in process model becomes a method “add_applicant_to_grop()” sent by object “Applicant data” to object “Group blank” presented on the interaction diagrams. As for UML sequence diagram, the description of an event in business process with its defined data structure in concept model can serve as basis for definition of the object, which is a node of its lifeline.

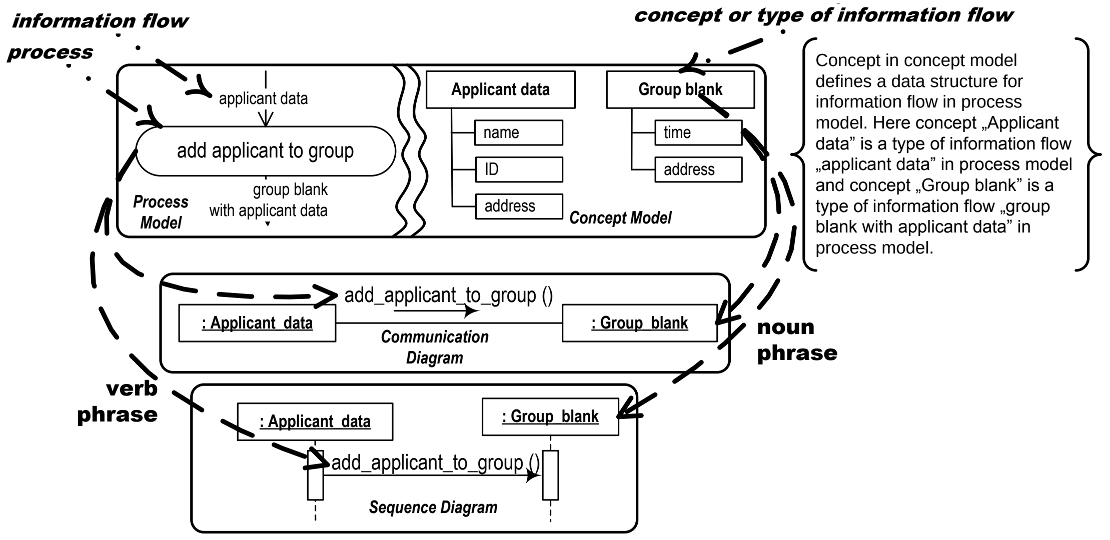

information flow
process
concept or type of information flow
applicant data
add applicant to group
group blank with applicant data
Process Model
Applicant data
name
ID
address
Group blank
time
address
Concept Model
Concept in concept model defines a data structure for information flow in process model. Here concept „Applicant data” is a type of information flow „applicant data” in process model and concept „Group blank” is a type of information flow „group blank with applicant data” in process model.
add_applicant_to_group ()
: Applicant_data
: Group_blank
Communication Diagram
noun phrase
verb phrase
: Applicant_data
: Group_blank
add_applicant_to_group ()
Sequence Diagram

Figure 2. Analysis of verb and noun phrases in two-hemisphere model and related object interaction.

The analysis of a verb phrase (see Fig. 2 [14]) makes it possible to suggest that the name of a business process has to be the base for the definition of a message of a sequence diagram to be performed by the object-receiver of this message. And if the name of the business process is defined in the form, where the first word is a verb, we can assume that the name of the exact message will be the same as the name of business process. According to the notation of the sequence diagram in [2], a message has the object-sender and the object-receiver of the message, which has to perform the action defined in the message. Direct transformation of graph of business processes into the graph of object communication defined in [9] solves the problem of identification of the object-sender and the object-receiver of the exact message by the application of several outlines of graph theory [15], where nodes of the graph of business processes have to be transformed into the arcs of the object communication and the arcs of the graph of the business processes have to be transformed into the nodes of object communication. The same assumption can be applied for the definition of objects in the sequence diagram – the object sender will be defined by an incoming arc of exact process in the model of business processes and the object-receiver will be defined by an outgoing arc of exact process in the model of business processes (see an example in Fig. 2 [14]). Therefore, the message defined to execute an exact process in the business process diagram will be sent by the object defined in the incoming arc of exact business process and received by the object in the outgoing arc of exact business process.

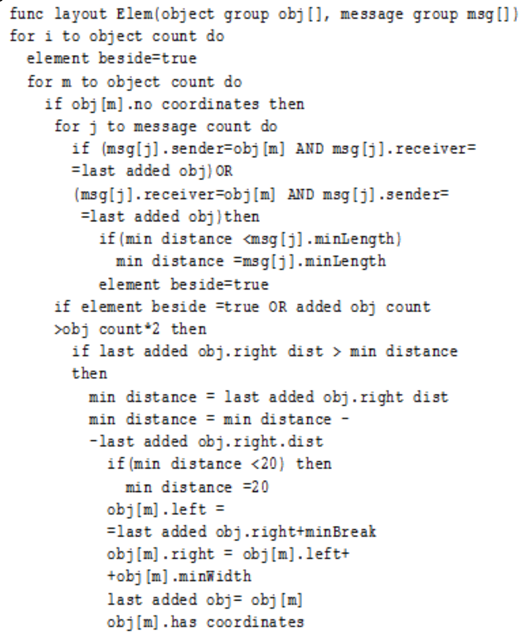
IV. THE TWO-HEMISPHERE MODEL DRIVEN APPROACH SUPPORT BY BRAINTOOL

BrainTool [16], developed by the researchers of the Riga Technical University, is a step forward in the area of automation of the modeling process. There exist a number of tools, which generate different UML diagrams. Some of them enable to define several elements of class structure based on a data presentation of the problem domain, e.g. Sparx Enterprise Architect or Rational Software Architect. Others generate the system model from the existing source code, to display the structure or the dynamic of the developed system, e.g. MS Visual Studio 2010. However, the problem of automatic generation of the UML diagrams from the formal and still customer-friendly presentation of the problem domain is not solved yet. Nikiforova et al. proposed to use BrainTool to generate UML class diagram from the two-hemisphere model [1]. Currently, the research group is working on a set of transformation rules for the generation of the UML interaction diagrams to built-in them into BrainTool and to expand a spectrum of the diagram supported by the tool. The essence of the transformations is described in the previous section. But the transformation provides only mapping of elements from a source to a target model. Layout of the model elements is another potential research problem to be solved to complete the task of supporting the automatic generation of the diagrams by BrainTool.

Diagram is a convenient way to represent information and is much more comprehensible than textual information. Although diagrams can be used to present complex and difficult problems, they must be semantically and syntactically correct and well layouted to give a desirable result. A good diagram needs to satisfy different criteria, among them aesthetic and layout criteria. General diagram criteria and specific UML diagram layout criteria have been studied by [17], [18], [19], [20] and others. All diagrams should comply with general graph layout criteria as a result from the theory of perception [17].

The UML communication diagram in the task of its layout can be accessed as usual graph, containing nodes connected by edges. Therefore, it is possible to use layout principles for usual graph layout. The UML sequence diagram, otherwise, is very specific in its visual presentation. All the objects are allocated horizontally at the top of the diagram and the lifelines are drawn vertically top-down. Therefore, the criteria for the UML sequence diagram should be carefully selected or even modified, so that they could be applied. E.g., one specific criterion for sequence diagram is correct sequence of messages, which is the meaning of this diagram. Poranen et al. [20] and Wong et al. [17] have identified the criteria specific for sequence diagrams, which are taking into consideration implementing the layout algorithm for the UML sequence diagram in BrainTool.

The layout algorithm tries to satisfy as many criteria as possible. It calculates the distance between the elements considering lengths of messages and class object names. Algorithm places elements as close as possible by taking into account the diagram flow (e.g., interacting objects are being placed beside if possible). The pseudo code of the layout algorithm implemented is presented in Fig. 3. The possible result of the transformation of the two-hemisphere model into the elements of the UML sequence diagram is shown in Fig. 4.



```
func layout Elem(object group obj[], message group msg[])
for i to object count do
  element beside=true
  for m to object count do
    if obj[m].no coordinates then
     for j to message count do
       if (msg[j].sender=obj[m] AND msg[j].receiver=
       =last added obj)OR
       (msg[j].receiver=obj[m] AND msg[j].sender=
        =last added obj)then
          if(min distance <msg[j].minLength)
            min distance =msg[j].minLength
          element beside=true
     if element beside =true OR added obj count
     >obj count*2 then
       if last added obj.right dist > min distance
       then
         min distance = last added obj.right dist
         min distance = min distance -
         -last added obj.right.dist
           if(min distance <20) then
             min distance =20
           obj[m].left =
           =last added obj.right+minBreak
           obj[m].right = obj[m].left+
           +obj[m].minWidth
           last added obj= obj[m]
           obj[m].has coordinates
```

Figure 3. Pseudo code of the layout algorithm.

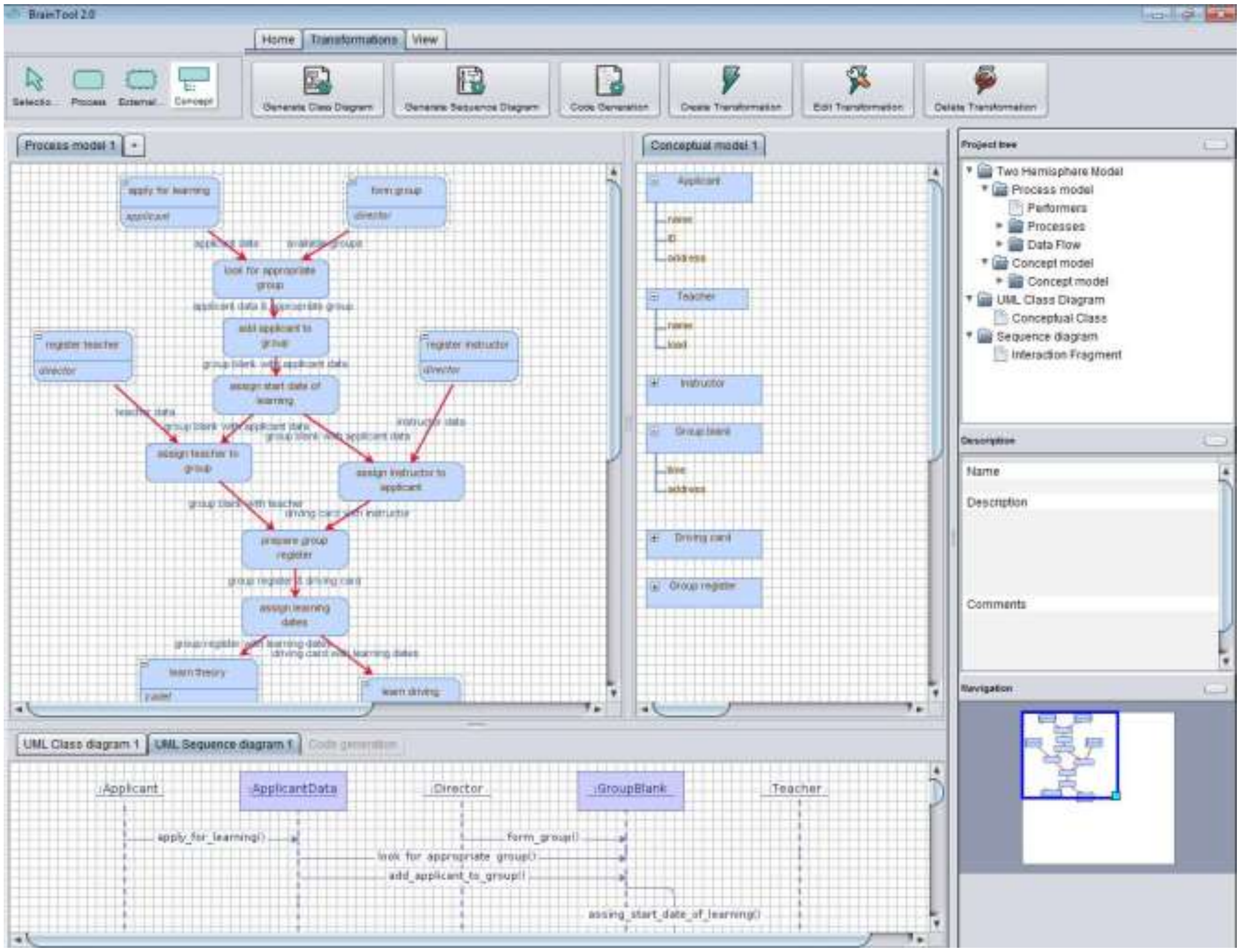


Figure 4. General view of BrainTool.

V. Comparison of the BrainTool with other UML compatible Tools

We have listed several tools offering creation of the UML interaction diagrams in Table 1, but they are mainly UML editors, where a developer creates all the diagrams manually with limited ability to generate new elements. Tools, like Sparx Enterprise Architect [21], Visual Paradigm [22] or Rational Software Architect [23] gives the ability to reflect to the existing UML diagram elements, if they are already created in other UML diagrams, but still, initially, these elements are identified manually.

Attempts to receive UML interaction diagrams from the requirements in natural language are one of the popular lines of research. For example, ReDSeeDS [24] supporting tool proposes linguistic analysis of system requirements and generates several elements of the UML sequence diagram, based on predefined format of requirements specification. But the tool has no graphical presentation of the resulting diagram and exports the result to Sparx Enterprise Architect.

On the other hand, Visual Studio supports the ability to generate the UML sequence diagram from the source program code. This is different direction from the approach offered in this paper and the tool can be interesting for comparison only in diagram presentation aspect, like as the diagram layout implementation, or export to other UML compatible tools.

There are several tools that provide automatic diagram layout, e.g., Borland Together [25] (not listed in Table 1) supports automatic UML sequence diagram layout, but uses lawless set of layout criteria). Sparx Enterprise Architect [21] is the tool that also provides automatic UML sequence diagram layout, however, it does not satisfy all the mentioned criteria of layout.

Thereby, we appreciate that currently abilities for the generation of the UML interaction diagram offered by the two-hemisphere model driven approach and supported by BrainTool are the most expansive, but we still have to refine the tool with additional functionality expected by users in popular UML editors.

TABLE I. COMPARISON OF BRAINTOOL TO OTHER TOOLS PROVIDING THE POSSIBILITY TO GENERATE THE UML INTERACTION DIAGRAM

Tool / Criteria	Visual Paradigm	Sparx EA	IBM RSA	Visual Studio	ReDSeeDS	BrainTool
Initial information for generation of the UML interaction diagrams	System req-ts & use-case diagram	System req-ts & use-case diagram	System req-ts & use-case diagram	Program code	System req-ts	Two-hemisphere model
Actors	Borrowed from use-cases	Borrowed from use-cases	Borrowed from use-cases	No	Automatically	Automatically
Objects	Manually	Manually	Manually	Automatically	Automatically	Automatically
Lifelines	Manually	Manually	Manually	Automatically	Automatically	Automatically
Operations	Manually	Manually	Manually	Automatically	Automatically	Automatically
Operation ordering	Manually	Manually	Manually	Automatically	Automatically	Automatically
Interaction frames	Manually	Manually	Manually	Automatically	Automatically	Automatically
Operation parameters	Manually	Manually	Manually	Automatically	Automatically	No
Links between objects (in communication diagram)	Manually	Manually	Manually	No	Automatically	Automatically
Transformation base	Linguistic analysis	Linguistic analysis	Linguistic analysis	Formal transformation text-to-model	Linguistic analysis	Formal transformation model-to-model
Model editor for initial information	Text editor	Text editor	Text editor	Text editor	Text editor	Graphical editor
Graphical representation of the UML sequence diagram	Yes	Yes	Yes	Yes	No	Yes
Graphical representation of the UML communication diagram	Yes	Yes	Yes	No	No	Not yet
Automatic layout	Not for UML sequence diagram	Lawless ordering of objects in the top of diagram	Not for UML sequence diagram	Yes	No	Yes
Export abilities to UML compatible tools	Has special export format	Has special export format	Has special export format	No	Yes (at least to Sparx EA	Defined by XMI and importable in the tools supporting the standard specification

VI. CONCLUSION

In comparison with the traditional software engineering development methods the model-driven approaches provide software development based on models. Models are system abstraction; they are the main artifacts, which are used on each development step. Automatic model transformations are used to design and develop software systems in a more comfortable and faster way. A transformation takes the model created on one level of abstraction and converts it to the model on another level of abstraction. Numerous languages and tools exist, which support this kind of development process. However, it is still not possible to automate software implementation, because there are several problems, which do not allow completing the model transformation.

The research object of this paper was the generation of the UML interaction diagrams, based on the two-hemisphere model. Both activities for that are being investigated: they are element identification from the problem domain and the visual representation (i.e., layout).

Thus, the contribution of the paper can be summarized as follows:

- A set of transformation rules for derivation of elements to present object interaction in terms of the UML diagrams are defined and implemented in BrainTool;
- A set of elements, which still are not transformable from the two-hemisphere model, is defined and allows the author to state the directions for the future research;
- An algorithm for the layout of the UML sequence diagram is developed and implemented, which pass the core requirements put forward to the object lifelines, messages and interaction frames.
- The tool supporting the transformations presented in this paper is compared to other tools giving an ability to create UML diagrams.

The main conclusions of the research are the following:

- The two-hemisphere model contains sufficient amount of information about the problem domain to identify a variety of the elements for object interaction presentation.
- It is possible to define all the required transformations in the formal way; moreover, they can be implemented by general purpose programming language.
- The layout of the diagram is a complicated task due to a large amount and diversity of the criteria that should be taken into consideration when placing elements in the diagram.

• A modeler cannot use convenient algorithms for graph presentation to layout the UML sequence diagram due to its specific structure; therefore, some unique method should be applied.
• The quality of the layout algorithm strongly depends on the complexity of the diagram itself.

The transformations and layout algorithm offered in this paper are implemented in BrainTool [16] in order to expand the functionality of its first version presented in [1] with respect to the modeling of the UML sequence diagram. Analysis of mapping abilities of the two-hemisphere model with the UML sequence diagram indicates an ability to refine notational conventions of the two-hemisphere model in order to increase a variety of the elements of the UML sequence diagram. This can be stated as a direction for a further research. Additionally, further research directions can include potential transformations from the two-hemisphere model to other types of UML diagrams, e.g., state charts, activity, etc.

ACKNOWLEDGMENT

The research presented in the paper is supported by Accenture Latvian Branch, project No. L7950 “Development of Model Transformation Tool Prototype”, and by Latvian Council of Science, No. 342/2012 "Development of Models and Methods Based on Distributed Artificial Intelligence, Knowledge Management and Advanced Web Technologies".

REFERENCES

[1] O. Nikiforova, K. Gusarovs, O. Gorbiks, and N. Pavlova “BrainTool. A Tool for Generation of the UML Class Diagrams”, Proc.: The Seventh International Conference on Software Engineering Advances, 2012, pp. 60-69.

[2] Unified modeling language: superstructure v.2.2, OMG. Available: http://www.omg.org/spec/UML/2.2/Superstructure. [retrieved 09/ 2013]

[3] G. Loniewski, E. Insfran, and Abrahao S. “A Systematic Review of the Use of Requirements Techniques in Model-Driven Development”. In: 13th Conference, MODELS 2010, Model Driven Engineering Languages and Systems, Part II, Oslo, Norway, pp. 213—227.

[4] O. Nikiforova and M. Kirikova, “Two-hemisphere model drivenapproach: engineering based software development,” in The 16th International Conference Advanced Information Systems Engineering, A. Persson and J. Stirna, Eds. BerlinHeidelberg: Springer-Verlag, LNCS 3084, 09.2004, pp. 219-233.

[5] C. Sibertin-Blanc, O. Tahir, and J. Cardoso, “Interpretation of UMLsequence diagrams as causality flows,” in Advanced DistributedSystems, 5th International School and Symposium. Heidelberg:Springer, LNCS, vol 3563, 2005, pp. 126-140.

[6] R. Alur, K. Etessami, and M. Yannakakis, “Inference of messagesequence charts,” in Proceedings of the 22nd International Conferenceon Software Engineering (ICSE). New York: ACM Press, 2000, pp. 304-313.

[7] S. Uchitel, J. Kramer, and J. Magee, “Detecting implied scenarios inmessage sequence chart specifications,” in Proceedings of the 9thEuropean Software Engineering Conference and 9th ACM SIGSOFTInternational Symposium on the Foundations of Software Engineering(ESEC/FSE’01). New York: ACM, 2001, pp. 74-82.

[8] B. D. Aredo, “A framework for semantics of UML sequence diagrams inPVS,” JUCS, vol. 8, no. 7, 2002, pp. 674-697.

[9] O. Nikiforova, Object Interaction as a Central Component of Object-Oriented System Analysis, (ENASE 2010), Proc. International Conference Evaluation of Novel Approaches to Software Engineering, WS Model Driven Architecture and Modeling Theory Driven Development, Osis J., Nikiforova O. (Eds.), SciTePress, Portugal, 2010, pp. 3-12.

[10] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen, Object Oriented Modeling and Design. New Jersey, Englewood Cliffs:Prentice-Hall, Inc, 1991.

[11] C. Larman, Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design, 3rd ed. New Jersey: Prentice Hall, 2005.

[12] P. Chen, “The entity relationship model – towards a unified view of data,” ACM Trans. Database Systems, vol. 1, 1976, pp. 9-36.

[13] O. Nikiforova, L. Kozacenko, and D. Ahilcenoka “UML Sequence Diagram: Transformation from the Two-Hemisphere Model and Layout”, *Applied Computer Systems*. Vol.14, 2013, pp. 31-41, doi: 10.2478/acss-2013-0004

[14] O.Nikiforova, "System Modeling in UML with Two-Hemisphere Model Driven Approach," Scientific Journal of Riga Technical University. Computer Sciences, vol. 21., 2010, pp. 37-44

[15] J. Grundspenkis, “Causal domain model driven knowledge acquisitionfor expert diagnosis system development,” in Lecture Notes of theNordic-Baltic Summer School on Applications of AI to ProductionEngineering, Kaunas, Lithuania, K. Wang and H.Pranevicius, Eds. Kaunas: Kaunas University of Technology Press, 1997.

[16] BrainTool. Availablet http://braintool.rtu.lv/ [retrieved 09/ 2013]

[17] K. Wong and D. Sun, On evaluating the layout of UML diagrams for program comprehension: IWPC 2005, 13th International Workshop on Program Comprehension, St. Louis, Missouri, USA. IEEE Computer Society, 05.2005.

[18] H. Eichelberger and K. Schmid, "Guidelines on the aesthetic quality of UML class diagrams," Information and Software Technology, vol. 51, no. 12, pp. 1686-1698, 2009.

[19] A. Galapovs and O. Nikiforova, Several Issues on the Definition of Algorithm for the Layout of the UML Class Diagram: 3rd International Workshop on Model Driven Architecture and Modeling Driven Software Development In conjinction with the 6th International Conference on Evaluation of Novel Approaches to Software Engineering, Beijing, China. SciTePress Digital Library, 06.2011.

[20] T. Poranen, E. Makinen, and J. Nummenmaa, How to Draw a Sequence Diagram: SPLST'03 Proceedings of the Eighth Symposium on Programming Languages and Software Tools, Kuopio, Finland. University of Kuopio, Department of Computer Science, 06.2003.

[21] Sparx systems, "Enterprise Architect". Available: http://www.sparxsystems.com.au/ [retrieved 09/ 2013]

[22] Visual Paradigm, "Generate Sequence Diagram from Use Case Flow of Events", May 2011. Available: http://www.visual-paradigm.com/product/vpuml/tutorials/gensdfromfoe.jsp [retrieved 09/ 2013]

[23] IBM, Rational Software Architect. Available: http://www.ibm.com/developerworks/rational/products/rsa/ [retrieved 09/ 2013]

[24] ReDSeeDS. Available: http://www.redseeds.eu/ [retrieved 09/ 2013]

[25] Borland a micro focus company, Borland Together, Available: http://www.borland.com/products/Together/ [retrieved 09/ 2013]

A Device For Electromechanical Braille Reading Digital Texts

Cicília Raquel Maia Leite, Davi Alves Magalhães, Pedro Fernandes Ribeiro Neto, Suellem Stephanne Fernandes Queiroz, Yáskara Ygara Menescal Pinto Fernandes
Department of Informatics
UERN (University of Rio Grande do Norte)
Mossoró, Brazil
ciciliamaia@gmail.com, davialvesmagalhaes@outlook.com, pedrofernandes@uern.br, suellem_stephanne@hotmail.com, yaskaramenescal@gmail.com

Abstract— **The social and professional inclusion of people with visual impairment is currently being sought enough. With accessibility is possible to integrate these people in order to provide equal conditions to them and thus make them an active part of society. Based in this theme, this paper proposes a prototype of an eletromechanical braille cell, which, with the use of an Arduino board, servomotors and software responsible for handling data, it is possible to represent in Braille information collected in the System Management Information Transit accessibility to Visually Impaired - TRANSITUS -V, making it behave like a human-machine interface for reading digital texts in braille.**

Keywords—accessibility; braille; technology; arduino; servomotors.

I. Introduction

In accordance with World Health Organization (WHO), there are approximately 160 million visually impaired people around the world, and at least 45 million of these individuals are completely blind [9]. Due to disability, these people have limited their basic rights as citizens. The situation is aggravated in digital media, where most of the visually impaired do not have access to special devices, or even help from trained professionals to help them in the usage of computers and other electronic equipments. Without the necessary resources, the person with a disability do not have the opportunity to fully utilize the phenomena that society experiences, such as social networks, in addition to competing at a major disadvantage to the jobs available that use of such technologies.

Organizations, states and society have turned their focus in ways to enable social and professional inclusion of people with visual impairment. Several devices have been and are being developed to allow the interaction of blind people with the computer. There are many prototypes in the literature with different proposals for cheapening and popularizing assistive technologies to blind people [1][5][7].

Grounded in this theme of accessibility and social inclusion of visually impaired, was designed a prototype that focuses primarily on the creation of a electromechanical Braille cell and implementation of a system composed of hardware and software that has the ability to interact with people totally blind, displaying in Braille informations obtained in TRANSITUS - V (Management of Information Transit Accessibility for the Visually Impaired), which has encouraged the development of new methodologies for the implementation of accessibility. This is an innovation if compared with other prototypes, once besides enabling the blind interaction with digital media, also enables their integration into the labor market and create conditions that these people become an active part of society.

Section II will describe the two main technologies involved in this project: Arduino and Transitus-V. Section III is a brief account of the Braille system and the use of servomotors in the construction of the prototype. In Section IV, the construction of the prototype is shown. In Section V the integration with Transitus-V is presented. Finally, Section VI provides a brief discussion about the obtained results.

II. Arduino plataform and system TRANSITUS-V

For being accessible, low cost and comprising hardware and software, the Arduino platform was used to preparation of this project due to its versatility and open source, ie it possible to reuse the hardware and the software libraries freely accordingly to the developer’s needs. Also, Arduino allows rapid prototyping of projects, which simplifies the manufacturing process by reducing the complexities inherent to the programming of the microcontroller and electronics prototyping.

The Arduino is already being widely used for the development of many projects focused on themes of social inclusion, which has encouraged the development of new methodologies for the implementation of accessibility. The TRANSITUS-V is a computer system with digital assistive technology that manages traffic information, developed in accordance with the W3C accessibility guidelines to facilitate the use and management of transit through people with visual impairments, with the use of shortcut keys, as well as special support for screen readers and voice synthesizers that increase the possibilities for use by persons with disabilities.

The system TRANSITUS-V, for having been done on a Web platform, requires no installation on the machines of those who use, each machine should only have access to the internet, and it’s compatible with most web browsers available in the market. However, the TRANSITUS-V needs to be hosted on a server that supports PHP and the MySQL database [6].

III. Braille system and Servomotors

The Braille system of reading and writing for the blind, was invented by Frenchman Louis Braille, influenced the society in the processes that led to the inclusion of these individuals. The Braille for its simplicity of reading and writing, was the bridge created between the blind and literature. Given the ease of use, the production of Braille content was encouraged, well as your teaching, spreading the method for worldwide.

With technological advancements, the Braille has been integrated into electronic devices allowing the interaction of the visually impaired with computers, text editors, internet, digital books, among other services. The example used in this project and has the servomotors, which are electromechanical devices that perform movements, in relation to its axis, in accordance with commands (control signals) determined. The device was a solution adopted for the project and is responsible for moving pins that make up a cell. Besides showing a cheap and easy to implement, since it easy to handle and has a library of software written specifically for use in conjunction with the the Arduino platform.

IV. Objective and case study

Through past difficulties for the visually impaired, the main objective of this work is the implementation and deployment of a system composed of hardware and software that displays Braille information extracted from a digital medium. The creation of the Braille cell electromechanical system adds the ability to interact with people who are totally blind, which is possible only through the web interface.

The prototype consists of parts of hardware and software to work together in translation and display of information acquired in TRANSITUS-V. The hardware part is formed by a plate Arduino BT, six servomotors, a button and secondary electronic parts, such as resistors and wires. Together, the six servomotors represent one braille character, the user can read a character string by advancing the read pointer by means of the button, as seen in Figure 1.

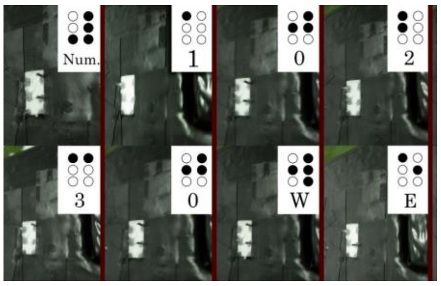


Figure 1. Braille representation of the data in the prototype

Each servo motor is responsible for moving one of the six pins that make up a braille cell. The position of the mechanical arm of each servomotor is determined by the micro controller to which it is connected, controlled angle values ranging between 0 º and 179 º. The characteristics of each servomotor Mystery Mini are shown in Table 1.

TABLE I. CHARACTERISTICS OF THE SERVOMOTOR MINI MYSTERY

Characteristics	Values
Quickness	60º at 0.12 seconds
Torque	0.7kg
Voltage	4.5v – 6v
Dimensions	1.98cm x 1.93cm x 8.4cm

Each servomotor has three wires: first, generally black or dark brown, which is the negative land should be connected to the circuit, the second, usually red, is positive, the third generally yellow in color, is attached to a PWM (Pulse Widht Modulation) port of the Arduino.

The hardware model of the Arduino platform used in the paper was the Arduino BT, chosen for having an integrated Bluetooth module to your hardware, which facilitates implementation. Another advantage of the model is to have six PWM digital ports, which allows the use of six servomotors, suitably representing a braille character. The features of the Arduino BT, is identical to the model Arduino UNO, with the exception of having an integrated Bluetooth module. The Arduino BT used has digital PWM ports 3, 5, 6, 9, 10 and 11 [8], and these ports are connected to the six servomotors.

Although the hardware platform Arduino usually have a power outlet dedicated to connecting other devices, a source of external power supply was used for the consumption of the servomotors, given its energy needs to be higher than what is offered at the output of the Arduino board. The diagram in Figure 2 describes the connection between the servo, Arduino BT and external power source.

To accommodate the servo motors and the Arduino board, a small box was built. In its lid six small holes, so that the servomotors to move small iron rods coupled to the blades make its surface appear in a character in Braille. The button used to move the cursor reading is powered by a 5V voltage obtained at one of the power ports Arduino board. The time between pressing the button and changing the character is about 0.1 seconds, according to the specifications of the servomotors seen in Table 1, making it very agile character exchange and enabling quick reading of the text displayed in the prototype.

The software part of the prototype is composed of the sketch that will run on the Arduino board, as well as a middleware responsible for brokering the acquisition of information. This middleware acquires the information from the database TRANSITUS-V and translates it into Braille to, finally, send them to skecth in the Arduino BT board. The sketch function is to coordinate the motion of the servomotors in accordance with the received information so that each character is represented correctly.

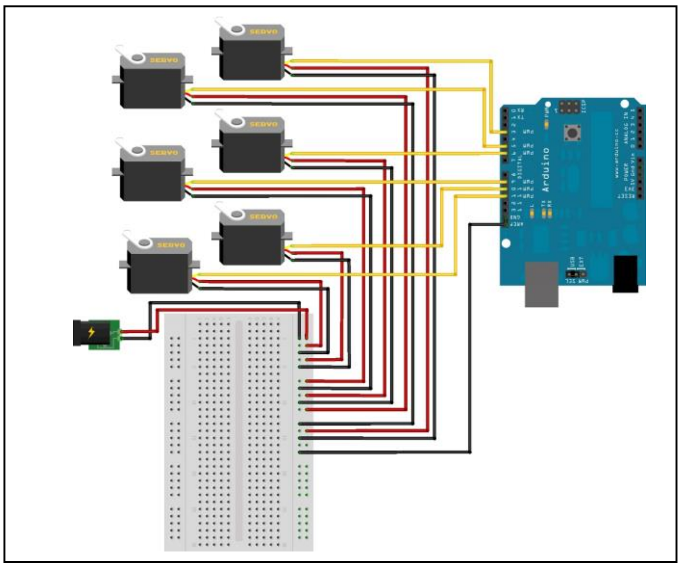


Figure 2. Diagram of connection between the servo and Arduino.

We opted for an application that communicates directly with the database because it simplifies its use by blind people. Even if the prototype can be adapted for use through the site, there are several steps prior to use which would be compromised. For example, it would be necessary that the blind user initiate the program from accessing internet browsing and the administrative area for, only then, have access to the information TRANSITUS-V. Direct access to the database reduces the steps required to use and consequently reduces the barriers that hinder the use of the system. So, all information is obtained by SQL queries.

After acquiring the specific data in TRANSITUS-V, the middleware sends character by character to the the Arduino board, previously converted to Braille system, and according to user demand. At first, only the first character is sent and immediately represented in the prototype. The user has the task of requesting the following characters one at a time, to middleware by pressing the button. Pressing the button causes the Arduino board send a request to the middleware, which is

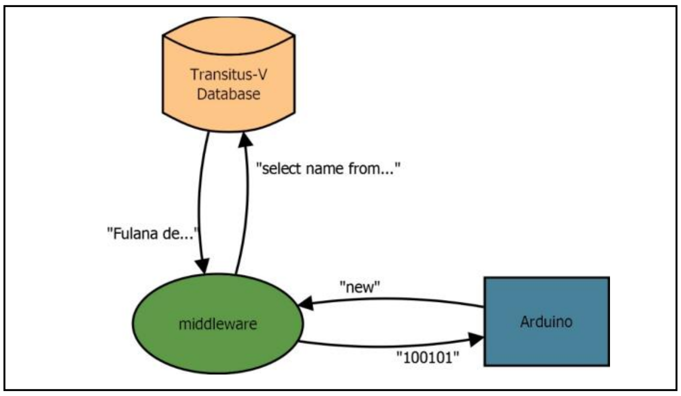
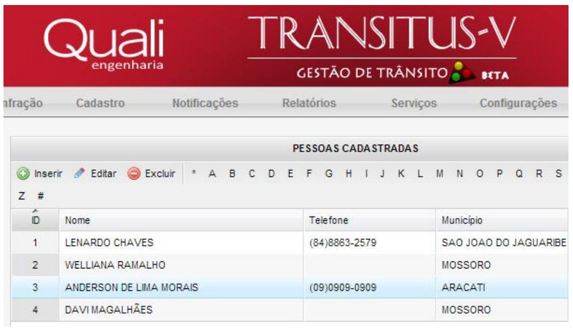


Figure 3. Diagram showing the exchange of messages between the Arduino, middleware and Database TRANSITUS-V.

done by sending the phrase “new” to the middleware. Upon receiving the request, the middleware sends the next character to the plate, and so on until they enclose the characters representative of the data obtained. Figure 3 describes how is this communication performed.

V. USABILITY AND PRACTICE

Figure 4 shows the names of customers registered in the system viewable on the web, with one highlighted in blue. The information is represented on the prototype, as seen in Figure 5. It is observed that the prototype represented correctly the desired information.



PESSOAS CADASTRADAS

ID	Nome	Telefone	Município
1	LENARDO CHAVES	(84)8863-2579	SAO JOAO DO JAGUARIBE
2	WELLIANA RAMALHO		MOSSORO
3	ANDERSON DE LIMA MORAIS	(09)0909-0909	ARACATI
4	DAVI MAGALHÃES		MOSSORO

Figure 4. Information obtained by persons web interface Transitus-V.

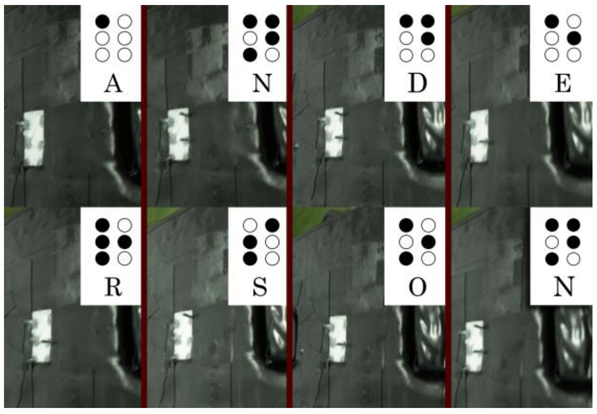


Figure 5. Braille representation of the data in Figure 4 prototype.

This work contributes not only with the realization of accessibility, but with the inclusion of visually impaired since the use of the prototype also allows the inclusion of these people in the labor market.

VI. FINAL CONSIDERATIONS

The prototype represented correctly in Braille information obtained in Transitus-V system. Further information can be obtained by simply writing functions and SQL queries to access them. However, the prototype does not have a navigation menu for the functions to access information, which is a barrier created by the low capacity of the displayed text. Because you can only represent one character at a time, creating a navigation menu is infeasible.

One difficulty encountered during the development of the prototype is that, given the angular movement of the blades of the servomotors, the pins do not rise or debase totally straight, which makes difficult the construction of smaller prototypes. Reducing the size of the blades was required.

This work is the result of a research project called Management of traffic information for the visually impaired, developed by the Software Engineering Group of the University of Rio Grande do Norte, which, since 2010, develops the web system TRANSITUS-V.

For future work, flip-flops can be used to build the braille cells, as seen in [1], giving to the system the capacity of representate a large number of characters. It is also suggested to create a shield for the Arduino platform representing Braille characters. This shield could pave the way for a family of accessibility projects, making it easier for the visually impaired and driving new research.

REFERENCES

[1] Braga, D. S. "Uma interface humano-máquina para leitura de documentos digitais por deficientes visuais". Escola politécnica de Pernambuco - Universidade de Pernambuco, 2010.

[2] Banzi, M. "Getting started with Arduino". http://goo.gl/Ue2Hr, First edition, 2009.

[3] Freedman, R. "Out of Darkness: The Story of Louis Braille", First edition, 1997.

[4] Godse, A. P., Mulani, A. O. "Embedded systems", First edition, 2009.

[5] Smithmaitrie, P. "Rehabilitation engineering – chapter 4: Analysis and design of Piezoeletric braille display". pp. 49-52. ISBN: 978-1-4302-3882-9. Apress, 2011.

[6] TRANSITUS-V. Official website of the TRANSITUS-V, http://les.di.uern.br/transitusportal/index.php/transitus-v.

[7] Wang, M; Roy, R. "Portable refreshable Braille display". Final Report for ECE 445, Senior Design, 2012.

[8] Wheat, D. "Arduino internals". ISBN: 978-1-4302-3882-9. Apress, 2011.

[9] World Health Organization. "World report on disabilities 2011". ISBN: 978 92 4 068521 5

Australia’s National Transition Strategy: first stage implementation report

Justin Brown
School of Computer & Security Science
Edith Cowan University
Perth, Western Australia
j.brown@ecu.edu.au

Scott hollier
School of Computer & Security Science
Edith Cowan University
Perth, Western Australia
j.hollier@ecu.edu.au

Vivienne Conway
School of Computer & Security Science
Edith Cowan University
Perth, Western Australia
v.conway@ecu.edu.au

***Abstract*—In June 2010 the Australian government introduced the National Transition Strategy (NTS), a mandatory requirement that all government websites in Australia would adhere to WCAG 2.0 Level A by the end of 2012 and AA by the end of 2014[1]. With the first deadline now past and many government websites remaining inaccessible, the failure of the NTS to date has raised questions in regards to its interagency support, community support and appropriateness of the NTS model. This paper explores the issues around the lack of NTS uptake to date: the choice of model, its implementation, and the lessons learnt and the likelihood of ultimate success as the 2014 deadline approaches**

Keywords- Australia, government policy, web accessibility, WCAG 2.0, conformance

I. INTRODUCTION

The introduction of the World Wide Web Consortium (W3C) Web Content Accessibility Guidelines (WCAG) 1.0 in 1999 was widely acknowledged as a significant step forward in the provision of online information to people with disabilities. While many countries adopted the guidelines into their policy and legislative frameworks, Australia took a more ad-hoc approach. With the release of WCAG 2.0 in December 2008 [2], Australia initially appeared to miss the importance of the web standard, with no significant changes to its web accessibility processes. However this changed significantly in 2010 when the Australian Government Information Management Office (AGIMO) released its National Transition Strategy, promising to make all Federal government websites WCAG 2.0 Level A compliant by the end of 2012, and Level AA by the end of 2014. State governments and territories within Australia also made similar commitments.

While the announcement was met with praise for the government’s approach to establishing a mandatory requirement on accessibility, the shift towards a uniform availability of accessible government information remained elusive. With the first deadline now past, it is important to reflect on the true impact of the NTS. In order to do so, it is first necessary to address the historical context of accessibility in Australia, the promise of the NTS, it’s approach compared with that of other countries and evaluate high usage government websites to determine the likelihood of the second NTS milestone being achieved.

This paper reports in part on an ongoing research project which is following the NTS through its implementation phase, the issues that led to the perception of failure thus far, the methods used in testing various websites to confirm WCAG compliance and key insights as to how web accessibility in Australia can be progressed despite the concerns over the current approach..

II. THE PATH TO A NATIONAL APPROACH

The primary catalyst for web accessibility being viewed as an important issue was the applicability of the Australian Disability Discrimination Act (DDA) [3] of 1992 as highlighted in the Maguire v Sydney Organising Committee for the Olympic Games (SOCOG) case. The case revolved around a legally blind man named Bruce Maguire who required ticketing and race information for the Sydney 2000 Olympic Games. Part of his complaint was that the information available on the official Olympic Games website was inaccessible, primarily due to the use of images without text descriptions. After taking all the arguments into consideration, the Australian Human Rights Commission (HREOC) came to the conclusion that SOCOG had “…engaged in conduct that is unlawful under section 24 of the DDA…”. [4].

As a result of the Maguire v SOCOG ruling, government policy began to acknowledge and incorporate the WCAG standard with brief references to the accessibility of online information requirements in the Federal Government Commonwealth Disability Strategy [5], but with most web accessibility policies being state-based, ad-hoc and largely implemented in a reactionary manner when issues in a particular website were raised [6]. However, the incremental acknowledgement of the importance around web accessibility and the release of WCAG 2.0 raised the possibility a specific strategy may be launched, with a number of speakers discussing the merits of a WCAG 2.0 strategy at the 2009 Gov 2.0 Roundtable on Accessibility for People with Disabilities [7]. The strategy was foreshadowed in an announcement in a media release by the Hon Lindsay Tanner MP in February 2010 that “Australians with disabilities will soon find it easier to access government information online” [8] with WCAG 2.0 selected as the policy requirements and that all government websites would be completed by 2015 [8].

III. THE NATIONAL TRANSITION STRATEGY (NTS)

The NTS was formally released on 30 June 2010 and declared to be a mandatory requirement and a formal endorsement of the Web Content Accessibility Guidelines (WCAG) version 2.0 for all government websites, superseding any policy that was previously based on WCAG 1.0. The formal release clarified the target dates by stating that all government websites must "…meet WCAG 2.0 Level A by December 2012…" and that all agencies were required to "…conform to WCAG 2.0 Level AA standard by December 2014" [9].

The introduction of the NTS heralded a significant shift in the implementation of web accessibility in Australia. The Government's Chief Information Officer, Ann Steward stated that the NTS "…sets a course for improved web services, paving the way for a more accessible and usable web environment that will more fully engage with, and allow participation from, all people within our society" [10]. The primary reasons as to why it was believed the NTS would make such a significant improvement to participation for people with disabilities was due to the NTS being the first time in Australia that a specific deadline had been set to implement web accessibility at a national level, that a formal strategy had been created and that WCAG 2.0 was acknowledged as the official Australian web accessibility standard.

The work plan for the NTS implementation was based on a three-phased approach:

Phase 1: Preparation - July 2010 to December 2010
Phase 2: Transition - January 2011 to December 2011
Phase 3: Implementation - Complete by December 2012 and December 2014

The first phase was for government agencies to take stock of their own websites, perform a conformance check, assess the website infrastructure, and assess their ability and risk in creating an accessible website. Phase 2 was designed to focus on accessibility training, procurement reviews and infrastructure upgrades, while Phase 3 was the implementation phase for accessible websites. The effectiveness of this approach hinged largely on the federal government agencies being subject to the Financial Management and Accountability Act [11], AGIMO will provide a reporting system, while agencies (those subject to Commonwealth Authorities and Companies Act) opting-in to the strategy are encouraged to report. The primary resource commitment given under the NTS is through the Web Guide [12] website with other resources to be created over time with the support of states and territories [1].

IV. IMPLEMENTATION PHASE OUTCOMES THUS FAR

Phase three of the NTS required implementation of the strategy in two parts, the first being the attainment of WCAG 2.0 Level A by the end of 2012 and then Level AA by the end of 2014. The initial research detailed in this paper indicates that the first stage of Phase three has not seen the NTS meet all of its accessibility goals.

The testing methodology included manual expert evaluation together with the use of three automated assessment tools, SortSite by PowerMapper [13], the Web Accessibility Toolbar (WAT) by the Paciello Group [14], and the WAVE extension for Mozilla Firefox by WebAIM [7]. SortSite was used to sample 2000 pages per site, while both the WAT and WAVE tools were used in conjunction with the manual expert assessments. The manual evaluation included 5 pages per site, typically being the homepage, contact us page, media pages and any pages featuring primary site information. It should be noted that there is some discrepancy between the manual testing and the automated testing results. The manual testing involved the 5 pages as stated and evaluated these pages against all WCAG 2.0 criteria. The automated testing while scheduled to check 2000 pages is unable, due to the nature and limitations of automated testing, to test more than about 35% of the guidelines effectively [5]. The automated tools were also used to test the 5 pages tested manually to verify and cross-check results.

Table 1 lists the largest of Australia’s federal government websites and their level of conformance to the first stage of the NTS implementation phase (as of end 2012).

TABLE 1: WCAG 2 LEVEL A CONFORMANCE FOR AUSTRALIAN FEDERAL GOVERNMENT AGENCY WEBSITES

Organization	**WCAG 2.0 Level A Pass**
Prime Minister's home page	No
Australian Government entry page	No
Department of Health & Aging	No
Australian Government Information Management Office (AGIMO)	Yes
Centrelink (now in Human Services)	No
Department of Education, Employment and Workplace Relations	yes (borderline)
Department of Immigration and Citizenship	no
Department of Infrastructure and Transport	Yes (borderline)
Australian Human Rights Commission	No
Australian Taxation Office	No
Employment services	No
Australian Job Search	No (borderline)
ABC Television (principally funded by federal government)	No
SBS Television (principally funded by federal government)	No

Medicare (now in Human Services)	No
Department of Finance (replacing AGIMO)	Yes
Department of Human Services (new site encompassing Centrelink, Medicare & Child Support)	No
Department of the Attorney-General	No
Department of Families, Housing, Community Services and Indigenous Affairs	Yes
Department of Broadband, Communications and the Digital Economy	No

Table 1 shows that only three sites actually passed the manual testing unequivocally, which is to be expected given that two of those websites belonged to the Australian Government Information Management Office, the owner of the NTS. The Department of Finance site is directly linked to AGIMO so is essentially run under the same structure. The third site which met WCAG 2.0 in the manual testing is that of the Department of Families, Housing Community Services and Indigenous Affairs which is one of the agencies in the reference group which was established to monitor progress. It is interesting to note that the Attorney-General's Department, and Department of Broadband are also in that reference group but whose websites did not pass WCAG 2.0 according to our testing. Two other departments have been defined as passing (with a borderline qualifier) as they had one or two issues which while a breach of the Level A guidelines, did not impact on site usability. One other site was a borderline fail, with some small issues that did impact on usability but would require minimal adjustments to achieve Level A. The organizations in Table 1 represent only a small selection of all the organizations which come under the mandate of the NTS on a national scale, however these are the mainline federal organizations and those which provide the most relevant test case for the NTS thus far. Mostly, they have the biggest budgets, the most staff and are the organizations that provide services and oversight to other federal and state entities.

While Table 1 provides a pass/fail evaluation for the websites examined according to WCAG 2.0, it should be kept in mind that this does not take into account the severity of the issues located, their frequency, or an analysis of the impact barrier upon people with disabilities. However, the NTS requires compliance with WCAG 2.0 to Level A by this time and does not allow for these additional criteria. Space restrictions within this paper mean the presentation of deeper analysis of automated and manual assessments is not possible here, though future publications of this research project will present such detail

V. DRIVERS FOR LEVEL A FAILURE

Looking at these representative Australian government agencies, what are some of the issues that have impacted on the lack of success of the NTS in the first part of it's implementation phase? Whilst this paper is not looking to cast a final judgment on the evolving NTS implementation, it does appear that whilst the NTS has lofty goals, it is lacking in specific details in terms of how to actually put web accessibility into practice, and how to assess it afterwards.

A. ASSIGNMENT OF RESPONSIBILITY

Perhaps one of the most glaring omissions in the NTS mandate is that of assignment of responsibility for implementation of each of the phases. The NTS documentation only ever refers to 'the Agency' or 'an Agency' but never to a specific role within these agencies, such as Chief Information Officer (CIO) or Chief Executive Officer (CEO). In comparison, the Canadian government's Standard on Web Accessibility names Senior Department Officials (SDO's) and CIO's [15] as being responsible for the implementation of their accessibility implementation. Under the U.S. Section 508, which links accessibility to government procurement, Chief Acquisition Officers and Chief Information Officers [16] are amongst those named as roles responsible for applying the requirements of the policy.

Information obtained from Australian federal government agencies in September 2011 as part of this research indicates that all of the agencies in this group have an individual who has responsibility for the accessibility of the website. However this does not explain the actual portion of a person's workload directly related to website accessibility. Survey data infers that the responsibility is often a small part of an incumbent's overall employment duties. Further information obtained in November 2012 shows that the number of agencies which have staff dedicated to the accessibility function has declined to the extent that one of the agencies identifies as not having anyone in the role and another is unsure. Whilst it might be 'everyone's' task to see that accessibility is applied at all levels of an organization, surely a senior role (ie CIO) needs to be named as being ultimately responsible [17].

One agency expressed the opinion that the website accessibility compliance is not the responsibility of just one individual but is built into the requirements, development and review process. As stated above when discussing roles, this may account for some confusion as to responsibility. If everyone is jointly responsible, then who is accountable when a website fails compliance?

While it would appear that Federal government agencies are working on improving the accessibility of their websites, it is apparent that there is much work still remaining. Some agencies have commented that they are aware they have not met the WCAG 2.0 A compliance deadline of December 31, 2012 deadline, but have decided just to continue to work toward WCAG 2.0 AA by December 2014. Due to the proximity of that deadline, it leads to the question of what, if anything will happen if they also fail to meet that timeline.

Some agencies state they are planning re-development of their website and that this re-development will address accessibility concerns. This would reinforce the common feeling in web development circles that it is easier and more cost-effective to re-design a website keeping accessibility in mind than to retrofit an existing site.

B. AUDITING METHODS AND TOOLS

The NTS documentation seems unclear in Phase three, Implementation, as to whether all sites need to be assessed upon reaching the 2012 deadline for single A compliance and then again in 2014 for double A compliance. The NTS Work Plan site [18] would seem to indicate that final compliance reports are to be completed at the end of the 2014 period. It seems that a compliance report at the end of 2012 would have provided agencies and the government as a whole with a useful 'dry run' of the final report due in 2014, perhaps highlighting issues in audit processes, methods and tools. The issue of auditing methods and tools is also a critical one, in that the NTS does not specify any particular method or tool beyond stating that "AGIMO will investigate whole-of-government automated conformance testing tools. It must be borne in mind, however, that automated testing tools can only interpret a limited range of criteria [5], which means that human judgment will also be needed in carrying out the tests. This will require staff skilled in web accessibility who can understand and apply the guidelines" [18].The Australian government's Web Guide is a little more specific in that it specifies that it is acceptable for most sites to test approximately 10% of their site (in terms of pages) and the sorts of items which should be tested, including home pages, contact details, feedback forms, search forms, online media and complete end-to-end process [19]. The Web Accessibility National Transition Strategy: Work Plan site appears to contradict this figure, stating that agencies "must ensure each web page meets WCAG 2.0 conformance requirements" [18]. Does this imply each page of those selected for assessment (say 10% of the site) or all pages in the site? It is this type of ambiguity, along with the somewhat loose language of the NTS and Web Guide documentation that allows for liberal interpretations of how agencies may perform their conformance reports. Terms such as 'At the very least', 'It is generally acceptable', 'Agencies may like to consider' and 'agencies are encouraged to complete' provide wriggle room for those agencies looking to take a minimalist approach to their accessibility commitments, at least in the short term. Whilst it may be expected that most agencies will do their best to implement the tenets of the NTS, the language of the documentation does not commit them to achieving the outcome but rather attempting to do so.

Survey results obtained from agencies about how they evaluate their websites provides further evidence about the confusion in evaluation and reporting. Some agencies have daily conformance checks for all new material, others state that they do not do any internal or external evaluation of the bnwebsite, with the rest falling somewhere in between.

C. INABILITY TO ENFORCE COMPLIANCE

Perhaps the most obvious issue with the NTS as it currently stands is its lack of enforceability. None of the NTS related documentation suggests any kind of penalty or censure for government web sites that do not achieve WCAG 2.0 AA compliance by the end of 2014. The Australia government's Web Guide indicates that once a federal site passes all the WCAG 2.0 AA success criteria it may use statements of conformance indicating they have met the 'five conformance statements of the WCAG 2.0'. Sites may also apply statements of 'partial conformance', such as where the site is heavily dependent on 3rd party providers who are not controlled by the agency or who do not come under the remit of the NTS. The final statement of the Web Guide in terms of conformance is that "where possible, agencies should aim to address accessibility issues as they occur" [19].

As far as available NTS documentation stands as of early 2013, the reward for an organization meeting NTS requirements is the ability to make statements of full or partial conformance against the NTS on their website. The apparent penalty for non-conformance is NOT being able to make such public statements. Whilst most federal agencies would relish the social capital and sense of achievement that would come from attaining NTS compliance, how this would be weighed against the time, money and ongoing effort such compliance would take remains to be seen [20].

VI. CONCLUSION

This paper has demonstrated that there are some key issues relating to Australia's National Transition Strategy that need to be addressed in order for people with disabilities to effectively use government websites. While the NTS has completed the first of two stages in its Implementation phase, an evaluation of essential federal sites within Australia has shown that, while the NTS has had a positive impact in progressing some accessibility awareness, it has yet to gain widespread traction within the government's web space. While it is commendable that Australia has taken a national approach in making government websites accessible and set specific accessibility targets unlike some other comparable countries [21]. However, the poor results of the first stage of the NTS implementation is largely attributable to a lack of resourcing and the need for a greater focus on consistent methods and toolsets [22].

The NTS provides the Australian government and the Australian population with the opportunity to proactively deal with the issue of equality of access for all things web. If this opportunity is squandered, digital citizens will continue to pursue their right to access online content and services through litigation and human rights avenues. Hopefully, the NTS and more than a decade of technical and policy development will obviate the need for further Maguire like cases to achieve web accessibility in Australia

REFERENCES

[1] [1] Australian Government Information Management Office (AGIMO) Australian Government web accessibility national transition strategy. City, 2010.

[2] [2] W3C Web Content Advisory Guidelines (WCAG) 2.0. W3C, City, 2008.

[3] [3] Australian Human Rights Commission D.D.A. guide: who does the D.D.A. protect? Australian Human Rights Commission, City.

[4] [4] W3C Web content accessibility guidelines (WCAG) 2.0. City, 2008.

[5] [5] Vigo, M., Brown, J. and Conway, V. Benchmarkng web accessibility evaluation tools: measuring the harm of sole reliance on automated tests. In Proceedings of the W4A 2013 colocated with 22nd International World Wide Web Conference (Rio de Janeiro, Brazil, May 13-15, 2013, 2013). ACM, [insert City of Publication],[insert 2013 of Publication].

[6] [6] Australian Government Web Guide: Accessibility. City, 2011.

[7] [7] WebAIM WAVE Toolbar 1.1.6. City.

[8] [8] Tanner, L. H. M. and Shorten, B. H. M. Dealing with government online to become easier for Australians with disabilities. Minister for Finance and Deregulation and Parliamentary Secretary for Disabilities, Australian Government, City, 2010.

[9] [9] Government, A. Web Guide : Accessibility. City, 2011.

[10] [10] Australia, M. A. Top 12 of 2012 #3 - The National Transition Strategy. Media Access Australia, City, 2012.

[11] [11] Deregulation, D. o. F. a. Financial Management and Accountability Act 1997 (FMA Act) Agencies. Department of Finance and Deregulation, City, 2012.

[12] [12] Government, A. Web Guide. City, 2013.

[13] [13] Powermapper software SortSite-Professional Edition. City, 2010.

[14] [14] Paciello Group Web Accessibility Toolbar for IE, 2012. Paciello Group, City, 2012.

[15] [15] Secretariat, T. B. o. C. Standard on Web Accessibility. City, 2011.

[16] [16] Administration, U. S. G. S. GSA Agency Roles and Responsibilities. City, 2012.

[17] [17] Bakhsh, M. and Mehmood, A. Web Accessibility for Disabled: A Case Study of Government Websites in Pakistan. In Proceedings of the Frontiers of Information Technology (FIT), 2012 10th International Conference on (17-19 Dec, 2012), [insert City of Publication],[insert 2012 of Publication].

[18] [18] Deregulation, D. o. F. a. Web Accessibility National Transition Strategy : Work Plan. City, 2010.

[19] [19] Government, A. Accessibility Conformance Testing. City, 2010.

[20] [20] Leitner, M. L., Hartjes, R. and Strauss, C. Web Accessibility Issues for the Distributed and Interworked Enterprise Portals. In Proceedings of the Parallel Processing Workshops, 2009. ICPPW '09. International Conference on. (Sept, 2009, 2009), [insert City of Publication],[insert 2009 of Publication].

[21] [21] Independent Hospital Pricing Authority. City, 2012.

[22] [22] Perrenoud, C. and Phan, K. Emergence of web technology: An implementation of web accessibility design in organizations. In Proceedings of the Technology Management for Emerging Technologies (PICMET), 2012 Proceedings of PICMET '12 (July 29 - Aug 2., 2012), [insert City of Publication],[insert 2012 of Publication].

Web Accessibility for Older Users: A Southern Argentinean View

Viviana Saldaño, Adriana Martin, Gabriela Gaetán, Diego Vilte
Department of Exact Sciences, Caleta Olivia
University of Patagonia Austral (UNPA-UACO)
Santa Cruz, Argentina
e-mail: vivianas@uaco.unpa.edu.ar, adrianaelba.martin@gmail.com,
ggaetan@uaco.unpa.edu.ar, dvilte773@yahoo.com.ar

***Abstract*—Older Web users are now facing one of the most difficult challenges of their lives. The Web changes every day and they cannot keep up with it. As older age comes, individuals experience gradual and fluctuating decline in capabilities. These physical impairments make usage of the Web even more difficult. Web accessibility is an area devoted to solve accessibility problems of disabled people. However, as older people suffer disabilities, although less severe ones, they can profit from Web accessibility solutions. In this article, we review some of the most common impairments that affect older Web users, we analyze how these impairments are considered by Web Accessibility standards, and explore different approaches that improve Web user interface. Finally, we introduce our ideas to overcome unsolved Web accessibility barriers for older users describing an experience carried out at our University in Argentinean Patagonia.**

Keywords - Web Accessibility; Older Web users; User Interface (UI);

I. INTRODUCTION

Most older adults experience age-related changes to their functional abilities (vision, hearing, cognition and mobility). These changes may complicate Web use [7], particularly for poorly designed sites. In Table I, we show some common functional impairments affecting older Web users, which we extracted from the literature review published by the W3C [21].

TABLE I. FUNCTIONAL IMPAIRMENTS AFFECTING OLDER WEB USERS

Ability	**Impact**	**Difficulties**
Vision	Screen Keyboard	1. Decreasing ability to focus on near tasks 2. Changing color perception and sensitivity 3. Pupil shrinkage and decreasing contrast sensitivity
Hearing	Audio Multimedia	4. Increasing inability to hear higher-pitched sounds
Motor skill	Mouse Keyboard	5. Slowness of movement, trembling
Cognitive	Overall Web use	6. Short term memory problems, concentration difficulties, distraction, change blindness

The study presented by Sayago and Blat [2] revealed that the accessibility barriers that had a more negative effect on the daily interactions of older people with the Web were remembering steps, understanding computer jargon and using the mouse.

Besides, from this study, we acknowledge that older Web users desire two conditions: independency and inclusiveness. Independency is the ability to use the Web on their own and inclusiveness is the need to interact with the Web using ordinary technology, as they do not intend to be different from the rest of users.

Another problem that older people have to face is social isolation [12]. Factors like diminished personal social networks, bereavement and health problems contribute to social isolation. Using the internet has significant value for elderly people, since it helps avoiding loneliness, boredom, helplessness, and decline of mental skills and it may increase the self-confidence, ability to learn, and memory retention.

Traditional communication technologies, such as the telephone, have played an important role in mitigating social isolation and supporting group gatherings. Also, the World Wide Web offers potential benefits for older adults, but its uptake is yet extremely limited.

There are many reasons why older adults do not use the Web [11]. Firstly, they tend to see the Internet as a tool to achieve functional goals such as bill payment, and not as a social or entertainment source [3]. Besides, they need an incentive to get and stay online [4]. It is often younger people who encourage technology use by older adults. Staying connected with geographically remote grandchildren is a major motivation for older adults in using technology (such as email, Web cams and Skype). An interesting finding was reported in [25], in which it is suggested that given the right trigger many older people (even those previously uninterested) will make tentative steps towards some technology. In this case, the trigger was a disaster, the “ash cloud”, which caused large scale disruption for air travel across Europe in 2010, and it motivated the need for computer usage.

Once older people are online they discover the advantages, such as being able to maintain existing social relationships and perhaps renew old ones that distance had precluded. Over two thirds of “silver surfers” say that using the Internet has improved their lives [5].

Other reasons for non-use of the Web include those involved with age-related impairments, such as the ones presented before in Table I.

In this paper, we explore different initiatives aimed at providing Web accessibility and usability properties for older users and some approaches to improve their Web interface experience [13]. Taking into account the state-of-the-art and the experience gained by our group while teaching computing to older people, we describe our ideas and show the improvements achieved during the delivery of the courses for elderly Web users. Since many fields are concerned on improving human-technology interaction, such as information retrieval and data mining, Human-Computer Interaction (HCI) and GUI, at this point, we have to clarify how we decided to face this work. We have been working for a while on accessible UI design to conform the W3C accessibility recommendations [26] [27]. Our knowledge gathered about UI design and Web Accessibility standards, permitted us to explore practical techniques to reinforce accessibility and usability and focus on the interaction between our seniors and the Web, using a real experience on Yahoo mail.

The rest of the paper is structured as follows: in Section II, we review Web accessibility standards and their relation with age related disabilities. Then, in Section III, we overview different useful approaches to improve older users' Web interface. After that, in Section IV, we describe an experience performed at our University and explain our ideas for improvement. In Section V, we introduce some discussion based on our experiences. Finally, in Section VI, we conclude and present some further work.

II. WEB ACCESSIBILITY INITIATIVE GUIDELINES AND AGING

The next few decades will see an unparalleled growth in the number of people becoming elderly compared with any other period in human history. The United Nations estimates that by 2050 one out of every five people will be over 60 years of age, and in some countries the proportion will be much higher than this [1].

There are some initiatives that provide advice addressing Web accessibility and usability for all people. As regards older users, many requirements are already considered by these initiatives.

The World Wide Web Consortium (W3C) Web Accessibility Initiative (WAI) [16] brings together people from industry, disability organizations, government, and research labs from around the world to develop guidelines and resources to help make the Web accessible to people with disabilities including auditory, cognitive, neurological, physical, speech, and visual disabilities.

Among these series of guidelines developed by WAI, widely regarded as the international standard for Web accessibility, are: Authoring Tool Accessibility Guidelines (ATAG), User Agent Accessibility Guidelines (UAAG) and Web Content Accessibility Guidelines (WCAG).

- The Authoring Tool Accessibility Guidelines (ATAG) documents define how authoring tools should help Web developers produce Web content that is accessible and complies with Web Content Accessibility Guidelines.
- The User Agent Accessibility Guidelines (UAAG) documents explain how to make user agents (Web browsers, media players, and assistive technologies) accessible to people with disabilities, particularly to increase accessibility to Web content.
- The WCAG documents explain how Web content can be made accessible for people with disabilities. The WCAG 2.0 [19] has twelve guidelines, grouped in four fundamental principles of accessibility: perceivable, operable, understandable, and robust. Each guideline is in turn decomposed in a set of success criteria, which are classified within three levels of conformance: A (lowest), AA, and AAA (highest).

Another WAI project, Web Accessibility Initiative: Ageing Education and Harmonization (WAI-AGE) project [17] analyzed the Web accessibility requirements of older Web users based on the research and investigation of many people.

WAI-AGE has identified that the existing WAI accessibility guidelines address the majority of requirements of older people for Web use [10]. It also identified that many Web designers and researchers are not considering the WAI guidelines when making recommendations about Website design for older people.

Although the guidelines developed by WAI were not written with older users' problems in mind, they provide solution to many of them. In Table II, we show the results of performing a matching analysis between most common older people accessibility barriers, presented before in Table I, and the corresponding guideline in WCAG 2.0.

TABLE II. OLDER WEB USERS DIFFICULTIES AND CORRESPONDING WCAG 2.0 GUIDELINES

Difficulty	WCAG 2.0 Guideline
1. Decreasing ability to focus on near tasks	1.4
2. Changing color perception and sensitivity	1.4
3. Pupil shrinkage and decreasing contrast sensitivity	1.4
4. Increasing inability to hear higher-pitched sounds	1.2 – 1.4
5. Slowness of movement, trembling	2.1 – 2.2
6. Short term memory problems, concentration difficulties, distraction, change blindness	2.2 – 2.4 – 3.2 – 3.3

We can see that the first three difficulties, which are visual impairments, are addressed by WCAG 2.0 in guideline 1.4. The fourth barrier, a hearing disability, is tackled by guidelines 1.2 and 1.4. The fifth difficulties, motor impairments, are addressed by guidelines 2.1 and 2.2. Finally, the sixth barriers, cognitive difficulties, are considered by guidelines 2.2, 2.4, 3.2, and 3.3.

This way, we could see that WCAG 2.0 guidelines meet all older Web users' requirements. The problem is that few Websites have been designed with these guidelines in mind.

III. DIFFERENT WEB SOLUTIONS THAT IMPROVE SILVER SURFERS' EXPERIENCE

Older people’s functional impairments are very different in type (vision, hearing, mobility, cognitive) and severity, and usually change over time. Thus, it is very difficult to specify a unique Web interface that meets the requirements for all of them [6]. So, the solution could be that each individual older user would be able to select the appropriate configuration by themselves.

There are some very interesting works related with this idea such as the IBM’s Web Adaptation Technology [9], which develops a browser extension that allows manipulating Web content by combining and applying a number of page transforms and adaptations according to user preferences without requiring Web designers and developers to rewrite their Web content.

Another tool is the Senior Citizen on the Web 2.0 (SCWeb2) Assistance tool [8], which is designed to assist older users as they use Web 2.0 content. For some users, dynamic content can be problematic due to the many updating components throughout the page, causing them hesitancy, stress, and frustration about unexpected situations. This tool provides help only when users require it, avoiding assistance and browsing the page in the usual manner when support is not needed.

There are many other solutions which provide Web accessibility not specifically oriented to older people. For example, Garrido et al. [24] propose improving Web accessibility in client browsers through interface refactorings. This approach is called Client-Side Web Refactoring (CSWR), it allows to automatically create different, personalized views of the same application. The refactorings proposed are compliant with W3C guidelines.

Besides, there are tools that allow users to change the way Web content is presented. GreaseMonkey [20] is a Firefox extension that allows writing scripts to alter visited Web pages. It can be used to make a Website more readable or more usable, Web applications can be modified by adding content and/or controls to them. For instance, Mirri et al. [23] describe GAPforAPE (GreaseMonkey And Profiling for Accessible Pages Enhancement), an augment browsing system based on GreaseMonkey, which allows Web users to set up their preferences at client side and thus modifying content on the browser interface. This application includes a profiling system and a client side content transcoding system, based on a collection of scripts. In order to enhance the accessibility of Web content and to provide the best adaptation to each user by meeting their needs and preferences, scripts allow the transcoding of Web pages, by modifying the CSS rules, the HTML DOM, and also other scripts which are used by them.

IV. EVALUATION OF OLDER USERS’ EXPERIENCE IN PATAGONIA

Since 2009, the National University of Patagonia Austral and the National Institute of Social Services for Pensioners (PAMI) have signed an agreement [18] for teaching computing, music, and theatre courses to older people.

These courses are taught twice a week and last three months. Computing courses are the most crowded, having about 20 pupils each.

Older people who assist to computing courses have expressed that they come to learn computing because they want to keep in touch with their families, with their grandchildren who live in other country regions.

Here, in Patagonia, distances between cities or towns are extremely long; besides, we are 1242 miles away from the capital city, Buenos Aires. Moreover, the weather is a critical factor, too. Winters are very long and cold, and strong winds blow. As a result, older people spend most of their time inside their houses, and they often feel lonely. Thus, getting online can have positive benefits for them. Tools like Email, FaceBook and Skype can empower older adults to stay connected with their friends and family.

In this study, the purpose is to find out which are the accessibility failures that the email’s Web interface has got and evaluate if a more accessible interface would allow older people to utilize it more frequently and without suffering frustration for not remembering how to use the application.

A. Experiment 1

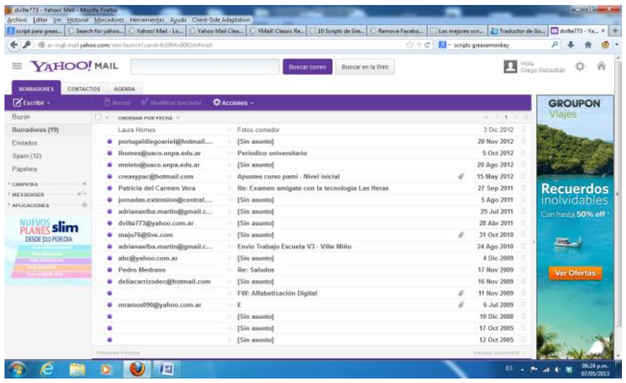
During the second half of 2012, teachers taught email classes. At the beginning of 2013, when computing classes started again, teachers noticed that most pupils did not use this communication tool. When asked for the reason of not using it, most pupils said that they did not remember how to use it, a few said that they were not interested in sending or receiving mails, and the rest, only some of them, said that they still used it. So, the purpose of this experiment is to investigate what accessibility difficulties has got the email’s Web interface design.

1) Participants:

Eighteen older adults ranging in age from 64 to 73 years old (eleven women and seven men) were recruited for this activity. All of them took computing courses between April and June of 2013 and also during the second half of 2012.

2) Materials:

For this experiment, we used Yahoo mail application (Figure 1) which was also used during email classes.



Laura Homes	Fotos comedor	3 Dic 2012
portugaldiegoariel@hotmail....	[Sin asunto]	20 Nov 2012
lhomes@uaco.unpa.edu.ar	Periodico universitario	5 Oct 2012
mnieto@uaco.unpa.edu.ar	[Sin asunto]	20 Ago 2012
creasypac@hotmail.com	Apuntes curso pami - Nivel inicial	15 May 2012
Patricia del Carmen Vera	Re: Examen amigate con la tecnologia Las Heras	27 Sep 2011
jornadas.extension@central....	[Sin asunto]	5 Ago 2011
adrianaelba.martin@gmail.c...	[Sin asunto]	25 Jul 2011
dvilte773@yahoo.com.ar	[Sin asunto]	28 Abr 2011
maju76@live.com	[Sin asunto]	31 Oct 2010
adrianaelba.martin@gmail.c...	Envío Trabajo Escuela V3 - Vilte Miño	24 Ago 2010
abc@yahoo.com.ar	[Sin asunto]	4 Dic 2009
Pedro Medrano	Re: Saludos	17 Nov 2009
deliacarrizodec@hotmail.com	[Sin asunto]	16 Nov 2009
	FW: Alfabetización Digital	11 Nov 2009
mramos090@yahoo.com.ar	E	6 Jul 2009
	[Sin asunto]	10 Dic 2008
	[Sin asunto]	17 Oct 2005
	[Sin asunto]	12 Oct 2005

Figure 1. Yahoo mail inbox.

It is important to highlight that the courses are taught in a 25 desktops Lab equipped with 15 LCD monitors of 19-inch and 10 LCD monitors of 17-inch, whose resolutions are WXGA 1366 x 768 and XGA 1024x768, respectively. Although changing terminals settings (font sizes and colors) is posible, the Lab is used intensively every day to adopt this practice as usual.

3) Procedure:

Usability testing with the think-aloud method was conducted [14]. The evaluations were pair-based because older people feel more relaxed and confident about their work. Each evaluation was recorded, in order to analyze participants behavior and comments.

4) Tasks:

Five tasks were proposed to explore the interface usability:

a) Read an email
b) Reply an email
c) Write a new email
d) Delete an email
e) Close user session

5) Results:

Of the 9 couples of participants, all could finish Tasks a) and c), 6 could not complete Task b), 2 could not conclude Task d) and 8 could not end Task e). These results are detailed in Table III.

TABLE III. RESULTS ACHIEVED BY OLDER USERS IN EMAIL USAGE *EXPERIMENT 1*

Task	Couples Error Ratio
a) Read an email	0/9
b) Reply an email	6/9
c) Write a new email	0/9
d) Delete an email	2/9
e) Close user session	8/9

From these results, we have found three problems throughout Tasks a)-e):

a) Problem 1: Advertisements

All participants complained about being distracted or even confused with the advertisements that appeared on the right side of the screen. They were afraid of clicking by error on these ads and causing an unexpected behavior of the email application, like closing, or losing the work being done.

b) Problem 2: Visual presentation difficulties

Besides, participants experienced other difficulties involving visual presentation of pages. Three couples of participants in Tasks a) and b) could not differentiate selected emails, because of light color contrast. Three couples of participants in Task a), three in Task b), and five in Task c) had difficulties in visualizing text because of font size, style, and inter-letter spacing. Also, 6 couples of participants in Task d) and 9 in Task e) made a great effort to distinguish available commands in menu bar.

c) Problem 3: Not understandable buttons

Participants also had trouble identifying buttons that represented email actions like "Reply" or "Forward". Eight couples of participants had difficulties identifying the button to conclude Task b), and 6 couples could not complete the task because of this problem. All participants had difficulties in Task e), remembering how to leave the application or "Sign Out", and only one couple could complete this task.

All the difficulties suffered by older users, are age-related issues like cognitive and visual impairment. Another factor involved is the lack of knowledge of technology and Web applications. Evaluating the WCAG 2.0 guidelines, we found that all these problems are considered within WCAG guidelines as we demonstrated before in Table II. Problems 1 and 3 correspond to difficulty number 6 detailed in Table II, which involves short term memory problems, concentration difficulties, distraction, and Problem 2 involves visual accessibility barriers shown as difficulties 1, 2, and 3 in Table II.

Hence, Yahoo email application is not compliant with this standard. However, this application provides solution to some of them, by setting appropriate configurations. But this is a very complex task to be performed by older users.

B. Experiment 2

The purpose of Experiment 2 is to evaluate an improvement to the email Website interface, which we developed to solve the problems found in Experiment 1.

In this improved interface, vertical banner ads have been deleted, and labels have been added for "Reply" and "Forward" buttons. Also, a button was added at the top of the form to allow users closing their sessions.

Figure 2 shows the modified interface of Yahoo mail inbox, including both adaptations: for Problem 1 vertical ads banner removement and for Problem 3 a button ("2" in Figure 2) labeled "Cerrar Sesión" to close user session, and the two labels "Responder" y "Reenviar" ("1" in Figure 2) for replying and forwarding respectively.

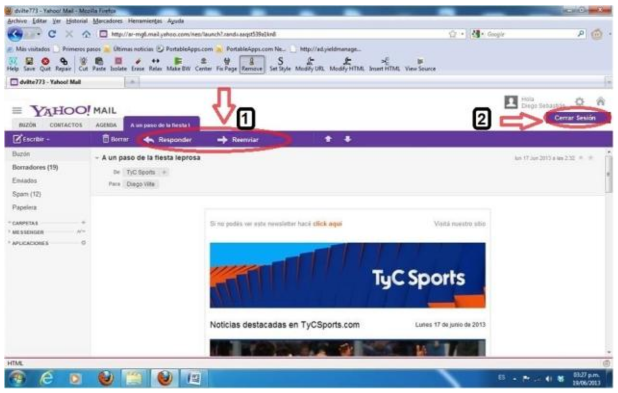


Figure 2. Yahoo mail inbox after interface improvement.

1) Participants:

Fourteen older adults ranging in age from 66 to 74 years old (eight women and six men) were recruited for this activity. All of them took computing courses during the first half of 2012, and now they are taking theatre but not computing classes. However, they were willing to participate in this experiment.

2) Materials:

We modified Yahoo interface by applying two adaptations [15]. One of them is a script for deleting vertical ad banners that we downloaded from a scripts repository and the other one is a script developed for us in JavaScript to solve problems with buttons.

a) Problem 1: Advertisements

Although this vertical banner ad can be removed, this was not a permanent solution and became an annoyance to older pupils. In order to give solution to this problem, we chose GreaseMonkey. There are many add-ins that provide a number of features for visual and navigational enhancements to Web pages, which may fill usability gaps for older users.

Figure 2 shows the modified interface of Yahoo mail inbox where the vertical banner has been deleted. This modification was achieved by the installation of a GreaseMonkey script, CleanUp 1.1 that we downloaded from the scripts repository [22].

b) Problem 2: Visual presentation difficulties

Here, there are solutions provided by the browser and also by the operating system. The browser (Mozilla Firefox) allows modifying default settings for font size and style, and the operating system (Windows 7) provides an Accessibility Center that allows improving visual presentation, mouse setting and color contrast.

c) Problem 3: Not understandable buttons

At this point, we did not find any GreaseMonkey script, which solves difficulties with buttons' understanding or 'Sign Out' explicit inclusion in the application interface. So, we developed a script named "Oldie 1.0" that added labels to "Reply" and "Forward" buttons and a button to allow users closing their sessions.

3) Procedure and Tasks:

The same as for Experiment 1, detailed in Sections IV.A.3) and IV.A.4) respectively.

4) Results:

Of the 7 couples of participants, all could finish Tasks a), c) and e), 1 could not complete Task b), and 1 could not complete Task d). These results are detailed in Table IV. In this experiment, Problems 1, 2 and 3 detected previously have been eliminated. A couple of participants could not finish tasks b) and d) because they did not remember how to perform those tasks.

TABLE IV. RESULTS ACHIEVED BY OLDER USERS IN EMAIL USAGE *EXPERIMENT 2*

Task Id	Task Description	Couples Error Ratio
a)	Read an email	0/7
b)	Reply an email	1/7
c)	Write a new email	0/7
d)	Delete an email	1/7
e)	Close user session	0/7

So, we conclude that this improved interface contributed to obtaining a better performance of older users and this will pay off in more confident users, who use email application more frequently and who are willing to go on learning new Web applications.

V. DISCUSSION

Many of the difficulties suffered by older Web users are already solved. However, as older people do not recognize their disabilities, they miss the opportunity to use the Web in a more comfortable way.

There are many accessibility tools provided by the operating systems and also by the Web browsers. But as they are classified as 'Accessibility Tools', most users believe that they are targeted to help people with severe disabilities that do not include the elderly.

Besides, there are some useful accessibility tools developed and available in Web repositories.

We have worked with some email accessibility requirements detected while teaching computing courses to older adults. Experiment 1 allowed for gaining a significant experience to develop our ideas, while Experiment 2 applied for testing these ideas on the field.

We found that some of the detected requirements could be solved by modifying the Web browser or the operating system configuration. Other requirements were accomplished by installing some scripts that provide the desired accessibility adaptations, like the scripts (CleanUp 1.1 and Oldie 1.0) we proposed and developed to solve Problems 1 and 3, respectively.

However, all these solutions require assistance from a computing specialist, or at least, from someone with the required skills, who must configure or install the appropriate add-ins.

Thus, we are working on a pragmatic research approach and applying an iterative incremental process to develop a tool that includes all the accessibility adaptations and allows older people select the appropriate configuration by themselves. Besides, this tool must be able to provide help to older users, who are not familiar with application concepts and hence avoiding hesitation and frustration. All this will contribute to increasing quality of life of our Patagonian older Web users.

VI. CONCLUSION

Older adults represent the fastest growing portion of the world's population. Most older adults have got some declines that affect computer use, as difficulties with vision, hearing, mobility or cognition.

The World Wide Web Consortium (W3C) has got some initiatives like Web Accessibility Initiative (WAI) and Web Accessibility Initiative: Ageing Education and Harmonisation (WAI-AGE), which provide solutions to many of the problems of older people. However, many Web designers do not consider WAI recommendations when designing Websites.

So, there are some approaches focused on improving Websites' accessibility. Some of them consist on Web adaptations that provide solution to a varying amount of accessibility issues.

In this article, we showed different solutions provided to solve distinct older pupils' requirements. However, from our experience, we must highlight two issues about these solutions: (i) they do not cover all needs and, (ii) they are not usable enough for elderly citizens. Due to these reasons, new solutions should be developed and these solutions must prevent older people having to get help from someone else who can configure or install suitable accessibility settings to grant our seniors one of their main wishes: “independence”.

As regards social requirements of our older students, our next goal is exploring difficulties experienced by them with social networks and finding appropriate solutions. This is a high priority requirement of our older citizens since our distant geographical situation and extreme weather conditions deprive them of enjoying many current activities that older people in other geographies can perform.

Acknowledgment

This work is supported by the UNPA project 29/B144 (Diseño y Evaluación de Portales Web).

References

[1] United Nations Department of Economic and Social Affairs Population Division, “World population ageing, 1950-2050,” UN, 2002, accessed 15th May 2013.

http://www.un.org/esa/population/publications/worldageing19502050/index.htm

[2] S. Sayago and J. Blat, “About the relevance of accessibility barriers in the everyday interactions of older people with the Web,” Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A '09), ACM, USA, April 2009, pp. 104-113.

[3] N. Selwyn, “The information aged: A qualitative study of older adults’ use of information and communications technology,” Journal of Aging Studies, 18(4), November 2004, pp. 369-384.

[4] E. Hartnett, S. Minocha, J. Palmer, M. Petre, S. Evans, C. Middup, K. Dunn, B. Murphy, T. Heap, and D. Roberts, “Older people and online social interactions: an empirical investigation,” The UKAIS International Conference on Information Systems (UKAIS), Worcester College, University of Oxford, March 2013.

[5] L. Gibson, W. Moncur, P. Forbes, J. Arnott, C. Martin, and A. Bachu, “Designing social networking sites for older adults,” Proceedings of the 24th BCS Interaction Specialist Group Conference (BCS '10), British Computer Society, Swinton, UK, September 2010, pp. 186-194.

[6] D. Sloan, M. Atkinson, C. Machin, and Y. Li, “The potential of adaptive interfaces as an accessibility aid for older Web users,” Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility (W4A '10), ACM, New York, NY, USA, April 2010, Article 35.

[7] V. Hanson, “Age and web access: the next generation,” Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A '09), ACM, USA, April 2009, pp. 7-15.

[8] D. Lunn and S. Harper, “Improving the accessibility of dynamic web content for older users,” Proceedings of the 2011 International Cross-Disciplinary Conference on Web Accessibility (W4A '11), ACM, New York, NY, USA, March 2011, Article 16.

[9] J. Richards and V. Hanson, “Web accessibility: a broader view,” Proceedings of the 13th international conference on World Wide Web (WWW'04), ACM, USA, May 2004, pp. 72-79.

[10] A. Arch, “Web accessibility for older users: successes and opportunities,” (keynote) Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A '09), ACM, USA, April 2009, pp. 1-6.

[11] A. Dickinson, M. Smith, J. Arnott, A. Newell, and R. Hill, “Approaches to web search and navigation for older computer novices,” Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '07), ACM, New York, NY, USA, May 2007, pp. 281-290.

[12] S. Pedell, F. Vetere, L. Kulik, E. Ozanne, and A. Gruner, “Social isolation of older people: the role of domestic technologies,” Proceedings of the 22nd Conference of the Computer-Human Interaction Special Interest Group of Australia on Computer-Human Interaction (OZCHI '10), ACM, New York, NY, USA, November 2010, pp. 164-167.

[13] S. Sayago, L. Camacho, and J. Blat, “Evaluation of techniques defined in WCAG 2.0 with older people,” Proceedings of the 2009 International Cross-Disciplinary Conference on Web Accessibililty (W4A '09), ACM, New York, NY, USA, April 2009, pp. 79-82.

[14] J. Dumas and J. C. Redish, “A Practical Guide to Usability Testing,” (1st ed.), Intellect Books, Exeter, UK, 1999.

[15] P. Brusilovsky, A. Kobsa, and W. Nejdl, “The Adative Web: Methods and Strategies of Web Personalization,” Springer, 2007.

[16] Web Accessibility Initiative, accessed 15th May 2013.

http://www.w3.org/WAI/

[17] Web Accessibility Initiative: Ageing Education and Harmonisation (WAI-AGE), accessed 15th May 2013.

http://www.w3.org/WAI/WAI-AGE/

[18] http://www.unpa.edu.ar/noticia/la-unpa-y-el-inssjp-firmaron-convenios-por-capacitacion-y-continuidad-del-programa-upami, accessed 10th June 2013.

[19] Web Content Accessibility Guidelines (WCAG) 2.0, accessed 15th May 2013. http://www.w3.org/TR/WCAG/

[20] Greasemonkey, accessed 15th June 2013.

http://www.greasespot.net/

[21] A. Arch, “Web accessibility for older users: a literature review,” W3C Working Draft, World Wide Web Consortium (W3C), May 2008, accessed 15th June 2013.

http://www.w3.org/TR/wai-age-literature/

[22] http://userscripts.org/, accessed 15th June 2013.

[23] S. Mirri, P. Salomoni, and C. Prandi, “Augment browsing and standard profiling for enhancing Web accessibility,” Proceedings of the 2011 International Cross-Disciplinary Conference on Web Accessibililty (W4A '11), ACM, New York, NY, USA, March 2011, Article 5.

[24] A. Garrido, S. Firmenich, G. Rossi, J. Grigera, N. Medina-Medina, and I. Harari, “Personalized Web accessibility using client-side refactoring,” Internet Computing, IEEE, 17(4), July-August 2013, pp. 58-66.

[25] L. Gibson, P. Forbes, and V. Hanson, “What can the ash cloud tell us about older adults’s technology adoption?,” Proceedings of the 12th international ACM SIGACCESS conference on Computers and accessibility (ASSETS '10), ACM, New York, NY, USA, October 2010, pp. 301-302.

[26] A. Martin, V. Saldaño, G. Miranda, and G. Gaetan, “AO-WAD: A Generalized Approach for Accessible Design within the Development of Web-based Systems,” Proceedings of The 7th International Conference on Software Engineering Advances, ICSEA 2012, IARIA, Portugal, November 2012, pp. 581-587.

[27] A. Martín, G. Rossi, A. Cechich, and S. Gordillo, “Engineering Accessible Web Applications. An Aspect-Oriented Approach,” World Wide Web Journal, 13(4), 2010, pp. 419-440.