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

 ISBN: 978-1-61208-304-9

(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 "

$$Availability = 1 - \frac{Down\,Time}{Measured\,Time} \quad (1)$$

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."

$$Accessability = \frac{number\ of\ Acks\ rmessages}{number\ of\ request\ messages} \quad (2)$$

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."

$$Successability = \frac{number\ of\ responsemessages}{number\ of\ request\ messages} \quad (3)$$

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.

 ISBN: 978-1-61208-304-9

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.

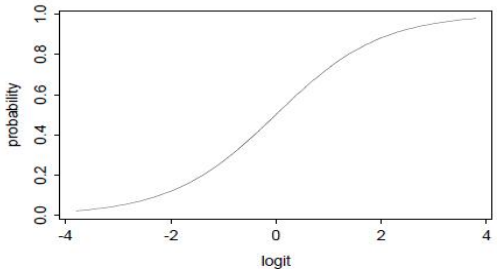

Figure 1. The logistic regression graph

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

 ISBN: 978-1-61208-304-9

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.

 ISBN: 978-1-61208-304-9

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

 ISBN: 978-1-61208-304-9

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]

 ISBN: 978-1-61208-304-9