FASSI 2019

The Fifth International Conference on Fundamentals and Advances in Software Systems Integration

ISBN: 978-1-61208-750-4

October 27 - 31, 2019

Nice, France

FASSI 2019 Editors

Mihaela Iridon, Candea LLC, USA
Chris Ireland, Open University, UK

FASSI 2019

Forward

The Fifth International Conference on Fundamentals and Advances in Software Systems Integration (FASSI 2019), held between October 27, 2019 and October 31, 2019 in Nice, France, continued a series of events started in 2015 and covering research in the field of software system integration.

On the surface the question of how to integrate two software systems appears to be a technical concern, one that involves addressing issues, such as how to exchange data (Hohpe 2012), and which software systems are responsible for which part of a business process. Furthermore, because we can build interfaces between software systems we might therefore believe that the problems of software integration have been solved. But those responsible for the design of a software system face a number of trade-offs. For example the decoupling of software components is one way to reduce assumptions, such as those about where code is executed and when it is executed (Hohpe 2012). However, decoupling introduces other problems because it leads to an increase in the number of connections and introduces issues of availability, responsiveness and synchronicity of changes (Hohpe 2012).

The objective of this conference is to work toward on understanding of these issues, the trade-offs and the problems of software integration and to explore strategies for dealing with them. We are interested to receive paper from researchers working in the field of software system integration.

We take here the opportunity to warmly thank all the members of the FASSI 2019 technical program committee, as well as all the reviewers. The creation of such a high quality conference program would not have been possible without their involvement. We also kindly thank all the authors who dedicated much of their time and effort to contribute to FASSI 2019. We truly believe that, thanks to all these efforts, the final conference program consisted of top quality contributions.

We also thank the members of the FASSI 2019 organizing committee for their help in handling the logistics and for their work that made this professional meeting a success.

We hope that FASSI 2019 was a successful international forum for the exchange of ideas and results between academia and industry and to promote further progress in the area of software systems integration. We also hope that Nice, France provided a pleasant environment during the conference and everyone saved some time to enjoy the charm of the city.

FASSI 2019 Chairs

FASSI Steering Committee
Chris Ireland, The Open University, UK
Hironori Washizaki, Waseda University / National Institute of Informatics / System Information, Japan
Keijiro Araki, Kyushu University, Japan

FASSI Publicity Chair
Ayman Aljarbouh, University of Grenoble Alpes (UGA) in Grenoble, France

FASSI 2019
Committee

FASSI Steering Committee

Chris Ireland, The Open University, UK
Hironori Washizaki, Waseda University / National Institute of Informatics / System Information, Japan
Keijiro Araki, Kyushu University, Japan

FASSI Publicity Chair

Ayman Aljarbouh, University of Grenoble Alpes (UGA) in Grenoble, France

FASSI 2019 Technical Program Committee

Frank J. Affonso, Universidade Estadual Paulista – UNESP, Brazil
Harvey Alférez, Montemorelos University, Mexico
Ayman Aljarbouh, University of Grenoble Alpes (UGA) in Grenoble, France
Keijiro Araki, Kyushu University, Japan
Doo-Hwan Bae, School of Computing - KAIST, South Korea
Imen Ben Mansour, University of Manouba, Tunisia
Dhouha Ben Noureddine, University of Carthage / University of El Manar, Tunisia
Silvia Bonfanti, University of Bergamo, Italy
Michael Franklin Bosu, Waikato Institute of Technology, New Zealand
Graeme Burnett, University of Glasgow/Enhyper Ltd., UK
Yudith Cardinale, Universidad Simón Bolívar, Caracas, Venezuela
Stephen Clyde, Utah State University, USA
Marcos Da Silveira, Luxembourg Institute of Science and Technology, Luxembourg
Marian Daun, University of Duisburg-Essen | paluno - The Ruhr Institute for Software Technology, Essen, Germany
Nitish Devadiga, Datarista Inc. / Carnegie Mellon University, USA
Francisco José Domínguez Mayo, University of Seville, Spain
Jorge Edison Lascano, Universidad de las Fuerzas Armadas ESPE, Quito, Ecuador
Leire Etxeberria Elorza, Mondragon Unibertsitatea, Spain
Ip-Shing Fan, Cranfield University, UK
Marie Farrell, University of Liverpool, UK
Peter Forbrig, University of Rostock, Germany
Lina Garcés, University of Sao Paulo, Brazil
Kevin Gary, Arizona State University, USA
Svetlana Gerbel, Hannover Medical School (MHH) / Medical Data Integration Center (MeDIC) / Hannover University of Applied Sciences and Arts, Germany
Atef Gharbi, LISI | INSAT, Tunisia
Hamza Gharsellaoui, ENI Carthage | Carthage University, Tunisia / Al Jouf College of Technology | TVTC, KSA
Fotios Gioulekas, Aristotle University of Thessaloniki, Greece
Afef Gueidi, University Tunis El Manar / Carthage University, Tunisia
Mohammad Mahdi Hassan, Al Qassim University, Buraidah, Saudi Arabia

Copyright Information

For your reference, this is the text governing the copyright release for material published by IARIA.

The copyright release is a transfer of publication rights, which allows IARIA and its partners to drive the dissemination of the published material. This allows IARIA to give articles increased visibility via distribution, inclusion in libraries, and arrangements for submission to indexes.

I, the undersigned, declare that the article is original, and that I represent the authors of this article in the copyright release matters. If this work has been done as work-for-hire, I have obtained all necessary clearances to execute a copyright release. I hereby irrevocably transfer exclusive copyright for this material to IARIA. I give IARIA permission or reproduce the work in any media format such as, but not limited to, print, digital, or electronic. I give IARIA permission to distribute the materials without restriction to any institutions or individuals. I give IARIA permission to submit the work for inclusion in article repositories as IARIA sees fit.

I, the undersigned, declare that to the best of my knowledge, the article is does not contain libelous or otherwise unlawful contents or invading the right of privacy or infringing on a proprietary right.

Following the copyright release, any circulated version of the article must bear the copyright notice and any header and footer information that IARIA applies to the published article.

IARIA grants royalty-free permission to the authors to disseminate the work, under the above provisions, for any academic, commercial, or industrial use. IARIA grants royalty-free permission to any individuals or institutions to make the article available electronically, online, or in print.

IARIA acknowledges that rights to any algorithm, process, procedure, apparatus, or articles of manufacture remain with the authors and their employers.

I, the undersigned, understand that IARIA will not be liable, in contract, tort (including, without limitation, negligence), pre-contract or other representations (other than fraudulent misrepresentations) or otherwise in connection with the publication of my work.

Exception to the above is made for work-for-hire performed while employed by the government. In that case, copyright to the material remains with the said government. The rightful owners (authors and government entity) grant unlimited and unrestricted permission to IARIA, IARIA's contractors, and IARIA's partners to further distribute the work.

Table of Contents

Design Model of a Training Simulator in Virtual Reality

Igor Petukhov
Department of Radio Technical
Volga State University of Technology
Yoshkar-Ola, Russia
email: Petuhoviv@volgatech.net

Andrey Glazyrin
Department of Radio Technical
Volga State University of Technology
Yoshkar-Ola, Russia
email: railot116@gmail.com

Liudmila Steshina
Department of Radio Technical
Volga State University of Technology
Yoshkar-Ola, Russia
email: Steshinala@volgatech.net

Dimiter Velev
Department of Information Technologies and Communications
University of National and World Economy
Sofia, Bulgaria
email: dgvelev@unwe.bg

Abstract— **This paper discusses problems that arise in the process of designing training simulators based on virtual reality. Virtual reality increases the performance of training due to immersion and realistic spatial objects. Unfortunately, there are problems associated with designing training simulators based on virtual reality. These problems are related to the performance of the environment in the context of effective user training. The paper presents a new approach to design a framework for a training simulator in virtual reality. Its key idea is to introduce basic principles for building of a two-level architecture using a user-centered design (on low-level) and object-closed design (on high-level). The low-level includes a modeling of the subject's orientation and the response of the environment to external influences. The high-level focuses on the specific of training scripts such as specificity of the operation or a detailed 3D model (visualization of target's operation through user interaction with the virtual environment). The data obtained can provide benefits to modeling training systems in virtual reality and for improving learning performance. The material presented can open new prospects for further research studies. It seems interesting to those who work in the field of usability engineering, training and human-computer interaction.**

Keywords- virtual reality; virtual environment; human-computer interaction; training simulator; virtual subjectivities; user-centered design; design framework component.

I. INTRODUCTION

The applications of Virtual Reality (VR) are becoming very popular in different fields of human activity. On one hand, there is a continued optimism in the growth of the immersive industry sector [1]. On the other hand, there are many opportunities in the contexts of communication and integration of human feelings and emotions in the Virtual Environments (VE) [2].

The greatest interest is simulation based training on VR (the system hardware and software are essential components of the virtual reality system), which affects the sense organs like in a realistic scenario of professional activity.

Despite the active progress of immersive and interactive technologies, some difficulties are still associated with certain restrictions. These problems include 3D interaction design in VR [3], creation of realistic 3D content such as physics and visual effects [4], unified techniques of interaction in the VE [5], the difficulties of geo-positioning and spatial relocation [6].

This paper covers actual issues linked with the analyses and description model of VR for training simulation, which takes into account the subject area and subjective user experience.

One of the most common applications of VR is simulation training in the different spheres such as medicine [7], astronautic science [8], education [9], industry [10], sports [11], military [12], games [13], building architecture [14], etc. Therefore, VR should reproduce a user's practical activity in the context of any task. At the same time, VR is safe for humans in comparison with the physical environment [15].

It was noted that the training of tasks that are performed in a three-dimensional space are better performed in VE [16], for example, memory training [17] or improving spatial thinking [18]. Moreover, perceptions of learning programs are becoming more effective in VE by increasing user motivations [19], modeling collaborative learning or other communication practices [20]. The greatest interest is training of movements and memorizing motor skills [21], such as simulations of accurate manipulations at atypical conditions for humans [8] [22].

It was shown that VE has an influence on psycho-emotional states and stress resistance [23]; thus, this one could activate the corresponding behavior like in the real world [24]. The analysis mentioned above shows the potential of VR in the context of increasing the effectiveness of learning and simulation training.

 ISBN: 978-1-61208-750-4

The rest of the paper is structured as follows. Section 2 describes the main problems of human-computer interaction within VR. Section 3 covers some related works in this area, summarizing the differences between characteristics and features of training simulation in VR. Section 4 mentions the mapping model, which is followed for the training simulator. Section 5 concludes this paper.

II. PROBLEMS

At the moment, the design of human-computer interaction within VR is centered on classical usability methods [25] that have been used in the Windows Icon Mouse Pointer (WIMP) - paradigm applications for a long time. At the same time, VR crucially differs from conventional desktop applications first of all by its deep psychophysiological action, a wider set of interaction techniques, and 3D contents [26].

Another key problem is related to the design of the immersion functionality. On one hand, there is an empirical correlation of immersion with hardware and software parameters of VR such as a frame rate, tracking a head rotation, audio, and interaction methods applied in the VE. On the other hand, a deep level of interaction can be explained by activation of similar structures in the brain, i.e. sensory stimuli as in the real world. Therefore, we face a problem of continuity between the subjective experience of the presence in the environment and the functional performance of the VR hardware [27].

It was noted that human performance is the basic element in VR because performance-based simulator-design guidelines include balancing perceived realism with simulator limitations, such as latency resulting from graphic and haptic renderings [28]. The problems of presence that affected humans in VR, such as user movement control, should be streamlined to enhance performance and reduce sickness [29].

The main principles of the complex processing of input information in VR were discussed [30]. This approach considers the user through the perception of the psycho-emotional model of the environment. On one hand, it is important to find a balance between rational reasoning and emotional reasoning because these factors integrate the human psychological state with VE [31]. On the other hand, there is the virtual subjectiveness [32], which affects consistency (mapping) between the cognitive-psychological level of the user's perception and the VR system [33].

Due to the problems mentioned, various research works and studies are focusing on finding out the components of visual immersion, including field of view, field of regard, and display size. Each element of visual immersion affects measurable user performance, understanding, and preference in a wide variety of VEs [34]. In this way, it is important to define what components affect the performance of which tasks [35].

However, there is a wide set of training simulator-based VRs that gives a good account of itself. These are VR simulators in medicine [36], education [37], communication [38], military [39], etc. So, let us consider how these problems are overcome. Based on these results, it is possible to describe the attributes and architectures (approaches) for designing the training systems in VE. It should be noted that the selection of parameters for the model of training simulators will be controlled by the specifics of the user-environment relations.

III. RELATED WORK

The Structural-Functional Design (SFD) overcomes the difficulties linked with the complex structure of the VR system and defines separated components, such as visual, behavioral and interaction characteristics. Each characteristic refers to the object's state inside the VR system and includes a set of parameters. For example, the visual level includes the rendering of the 3D content after the process of a user's interaction with the environment. The behavioral characteristic defines the actions of objects in VE and the interaction between 3D objects.

In the context of training, the design model finds out the components that may have a strong impact on the modeling of a realistic training simulation. The methodology formalizes the process of VR interface into two phases, which describe levels of abstraction, and breaks down the phases into components [40]. The high-level phase defines the conceptual feature of the environment (the target of training, methods simulations); at the same time, the low-level phase guides details of human interaction, rendering of 3D objects, behavior of the environment, etc.

Consequently, SFD helps to unify around the structure of the VR system, defines the components of the systems, and finds out the target and features of components. In practice, this methodology uses the Virtual Reality Interface Design (VRID) model [40], TRES-D [41] and other examples [42-44].

Unfortunately, the mentioned model focuses to a greater extent on technical details and ignores the specifics of participants. This conceptual framework may help to plan a design process or represents the operational behavior of the system. Therefore, it is important to consider other examples of the model of VR, which takes an active part in the interaction and communication with the user.

The Communication-Information Design (CID) suggests considering a training environment like an active subject of communication with the user [45]. For that reason, the mentioned environment contains a decision support system based on Artificial Intelligence (AI) that concentrates around avatars (virtual human being) and virtual surroundings.

The typical illustration of CID is the so-called Virtual Human Project (VHP) [46]. The goal of VHP is to create realistic virtual humans to increase the effectiveness of the communication information procedure of interaction between users and avatars. In this case, the user is a concurrent part of the training environment and active object in VE.

Conceptually, the virtual humans or avatars should include three nested layers that make up the mind the agent thinks with (cognitive layer), the body the agent acts with (virtual layer), and the world of the agent (simulation layer) [46]. Each layer is the set of components that extend features

 ISBN: 978-1-61208-750-4

of avatars and includes verbal speech, body gesture, and actions the character performs, for example, walking.

For training simulation, the approach mentioned may help to design the environment for cognition and emotion modeling of the user's condition. In practice, it is training in VR such as tactical questions in military or cultural immersive training [47], commutative capacity [48], and crowd simulation [49]. The specific feature of the communication–information approach is modeling virtual humans for interaction with the user through speech and gesture.

The Object-Closed Design (OCD) focuses on detailed implementation (visualization of granule operation's component, pressure feedback, quality of movement) of the complex manipulation in a variety of fields such as medical [50], handling operations [51], engineering [52][53], system of telepresence [54][55], etc. This approach includes monitoring the system in real time. In this case, the environment should be reacting on each event that appears after the user's manipulation, 3D object's interaction, the end of a fixed period, etc.

Therefore, VR should reproduce a user's practical activity in the context of any task. Indeed, the user is key to the system's component; at the same time, the reaction of the environment is more important. The user is defined as a secondary member and a concurrent element to perform any task. The communication between the training environment and the participant is executed through object-closed manipulation. For example, in the medical field, there is pressure on the special mannequin, imitation of elasticity and feedback of rendering a 3D view of anatomical structures [50].

The object-closed approach may help with detailed modeling of task execution. Unfortunately, this model disregards the significance of user's attribute such as motility, psychophysiological specificity, subjectivity, and experiences.

The User-Centered Design (UCD) models a training environment that consists of users (humans) as the most important items in interaction with virtual content through equipment. For that reason, the user is no longer "a black box" because this one may *be considered* like an object with previous experience or psychophysiological specificity. It was noted that human performance is related to the quality of the VE (level of immersion, self-explanatory navigation, ease of interaction with 3D object, etc.). At the same time, it has shown the positive and negative impact of VR on the health of humans [56]. Therefore, it is important to extract a human feature, which affects the performance of the environment. For example, in the Conceptual VR Model (CVRM), the user handles effectors (shell, fixture, appliance) from VR, which reduce feedback in the form of sensory stimuli. Consequently, for correct modeling, UCD finds out the mapping of the virtual effectors and the perceptual system of the participant. So, the visual perceptual system is linked with visual display such as orientation in time and space [57].

Conceptually, there are three independent main parts of the system, such as the environment, a user and a mediator. The mediator integrates the user with VE through Virtual Subjectivities (VS) [53]. The VS includes reminiscence about the surrounding medium and subjective experiences in the context of the psychophysiological-cognitive patterns that become active in the same situations as in physical reality. The mediator appears in the form of scale perception, orientation, action, etc. The UCD does user an active actor in the scheme of training systems because the virtual model combines human perception and dynamic spatial content. Unfortunately, the border between the user and the VE remains diffuse in this model. The mediator is a key component needed in defining the factors that support the performance of the training simulation in VR.

Table 1 summarizes the differences between the characteristics and features of different model designs. As the table indicates, each approach brings significant challenges in modeling the training environment. For example, CID fits collaborative training or face-to-face communication, but it is unlikely to be used in an illustration of surgical operation. UCD, for example, does not completely reflect the specific quality of the operation, but it probably allows to include the virtual subjunctives in the process of simulation.

TABLE I. PREVIOUS RESEARCH AND DEFINITIONS OF THE DESIGN MODEL OF TRAINING SIMULATION IN VR

Name	***Framework and design model***	***Type of training***	***Central elements***	***Key features***
SFD	Tanriverdi V., Jacob R. J. K. VRID 2001 [40], Molina J. P. et al. TRES-D 2006 [41], Cochrane T. et al. DBR 2017 [43].	none	The visual, behavioral and interaction characteristics	Defines components of the systems; finds out target and features of components
CID	Kenny P. et al. VHP 2007 [46], Prange A. et al. MDS 2017 [48], Ulicny B., Thalmann D. Crowd simulation 2001 [49]	Collaborative training, communication, crowd training, cultural interchange.	The cognitive level and AI, model of avatar (verbal speech, body gesture, and actions the character performs)	Creates a realistically virtual human to increase the effectiveness of communication–information procedure of interaction between users and avatars
OCD	Çakmak H. K., Kühnapfel U. KisMo 2000 [50], Pürzel F. et al. 2013 [51], Stoll	Modeling of granule operation's component, pressure feedback, quality of movement	The reaction of VR on actions of user (pressure, feedback, imitation of elasticity and	The special mannequin or a detailed 3D model. Visualization of target's operation through user

Name	*Framework and design model*	*Type of training*	*Central elements*	*Key features*
	E., Wilde M., Pong C. 2009 [54]	and etc.	etc.)	interaction with VR.
UCD	Stanney K. M., Mourant R. R., Kennedy R. S. 1998 [56], Latta J. N., Oberg D. J. et al. CVRM 1994 [57], Parés N., Parés R. 2006 [32]	Modeling of training simulator includes user experiences , characteristics, and psychophysiological-cognitive patterns.	The user handles effectors (shell, fixture, appliance) from VR and reproduces feedback in the form of sensory stimuli	Model of mapping virtual element's and correct user's perception.

It is necessary to emphasize that current models are linked with targets of training simulation and use different architectural components. The most interest brings UCD and OCD approaches' focus on the subjective perception of the environment and VE's reflection on input user's action. In the next section, the extended model of training systems based on UCD and OCD will be discussed.

Immersion or presence is a critical attribute of VR [58]. Immersion is the state of mind of an individual where he or she excludes the outside world and is totally focused on experiencing another world [59]. It was shown that the immersion appears in the form of cognitive and perception components of user's subjectivities [60]. On the one hand, immersion influences the performance and quality of an executed task [61][62] through correct selection and specification of spatial elements. In this context, the 3D content and property elements of VR are important attributes of the presence. Especially, the important role of physical laws [63], velocity [59][64], collision and occlusion [65] were shown.

There is a set of properties of VR devices that affect presence, for example head rotation [66], tracking system [67], screen resolution [68], and rendering [69]. Moreover, the empirical result found in [70] confirms the requirement for the presence of the following parameters: frame rate, tracking head rotation, sound, and technique of interaction.

The relation between the correct properties of spatial objects and any parameters of devices remains an open discussion. This problem has been considered through different schemes, for example, human reaction and subjectivities mapping.

Subjectivities mapping attracts the most interest because this approach defines two additional and important cues for the understanding of the psychological impact of VR. These two cues are the physical interface (any manipulation of devices based on the movement of the user) and the logical interface (any rendering or view's feedback after the movement of the user). Then, the virtual subjectivities impact on the environment itself seem to be a mapping or correct association between the user movement and the view rendering. Unfortunately, the approach mentioned is needed in the definition of mapping elements. At the same time, the elements are key to understanding the principles of modeling the training environment in VR. In the next section, we will discuss the mapping elements based on the training requirements and the framework for designing a training environment in VR.

IV. THE MAPPING ELEMENTS OF TRAINING SIMULATION

The sequence of human actions in a VE was shown [71]. Firstly, the person orients himself/herself in the VE and, after that, he/she interacts with the VE. We believe mapping elements might include a set of grouped human actions based on the priority for human perception inside the VE.

For this reason, the Queuing Network-Model Human Processor (QN-MHP) may help to describe the process of human perception through the functioning of the sensory-motor system based on three layers (sensory, cognitive and motor) [72]. Therefore, human actions are associated with ordered sensory-motor reactions. Indeed, the person perceives visual information through the sensory layer (sensory analysis). The visual information activates previous experiences from the human knowledge (the database of knowledge). Finally, the motor program is reproduced in the form of actions and manipulation (motor program).

These assumptions about the process of human perception and mapping elements may have a strong impact on modeling training systems. On one hand, the mapping in the VR system in context of human knowledge (the database of knowledge) from QN-MHP may include human perception of VE in form (distance = scaled, rotation = viewing angle, lighting = visual effects, sound = audio effects) and the simulation of behavior for the environment based on previous user experiences from real situations such as (physics laws = correct rendering 3D-content, tracing = moving reaction, fitting = distance reacting).

On the other hand, for modeling of the specific process in form of focused actions should be included components from human perception of VE and the simulation of behavior for VE. We believe this combination is a high-level model for object-closed design. It is focused on specific training simulation. The relation between mapping and design levels for the training simulation is shown table 2.

The sensory-motor activation in training simulation with mapping model may help to understand the relation between VE and the functioning of the human perception. For this reason, each perception layer may be linked to virtual subjectivities, which include logical interfaces, physical interfaces and mapping.

The logical interface is responsible for visual effects in context of virtual subjectivities. In this way, human perception in the form of sensory analysis is related to the logical interface through visual feedback. The visual feedback perceives from the database of knowledge «Conceptual model» inside the cognitive layer. The extracted situational model may be corrected according to the current situation. Accordingly, the synchronization of previous user experiences is triggered.

 ISBN: 978-1-61208-750-4

TABLE II. THE CONCEPTUAL SCHEME OF MAPPING ELEMENTS OF TRAINING SIMULATION

High-level (OCD)	**Object-closed modeling**			
	Execution a task: The logic of application with modeling of different scripts and important of components (the imitation of workflow, operation's quality, precedence, and time delay).			
Output:	Logic interface (correct rendering of VE as feedback from physics interface)			
Low-level (UCD)	**User-centered modeling**			
	Orientation		Imitation	
	The mapping elements			
	Distance	Scaled	Moving	Time correlation
	Rotation	Viewing angle	Tracking	Moving reaction
	Lighting	Visual effect	Fitting	Changed distance
	Sound	Audio effect	Physics laws	Correct rendering
Input:	Physics interface (manipulation with virtual devices: Head-mounted display, virtual glove, tracking, joysticks and etc.)			

At the same time, the corrected model influences to choose motor action in the form of "motor reaction". Finally, this motor reaction converts to muscle efforts through the physical interface. The mentioned steps are summarized in Figure 1.

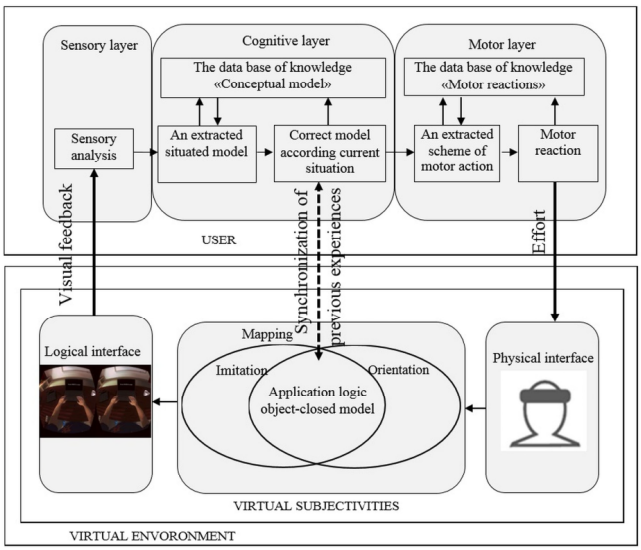

Figure 1. The user's role in training simulator based on mapping model

The mentioned model is focused on human reactions, which are related to virtual subjectivities through synchronization of previous user experiences. Therefore, the abstract database of knowledge «Conceptual model» needs great numbers of training situations for effective training. It reminds us of training a set of examples for Artificial Neural Networks (ANN).

We do not know the deep principles of brain learning. At the same time, there are different primitive models of the human brain such as ANN. This models show better results than human beings in some tasks such as classification or image recognitions. For that reason, we should make an analogy about ANN and «Conceptual model» from the mapping model. The ANN gets many various pieces of data for training, and then a training simulator based on «Conceptual model» may be considered as the generator of nonrecurring learning situations. Those situations may help to overcome the problems that are linked with the satiation of the database of knowledge «Conceptual model».

V. RESULTS AND CONCLUSION

In this paper, we identified the contemporary approaches to the design model for the training simulator in VR. It was noted that there is a relationship between the type of training and the design model of training simulator in VR. The greatest interest is in a design model based on OCD and UCD. Both approaches are perspective in different fields of training process. These approaches offer to focus on a detailed process of task execution is the same as integrating the user into the workflow. We believe in a central role of human reactions in the training process based on the mapping model.

The mentioned approach for training simulator based on VE allows us to define a design framework, including two design levels. The low-level UCD paradigm focuses on the human reaction, simple actions and perception. This level includes mapping logical and physical interfaces.

On one hand, the main target is a correct adjustment of mapping using scaled setting, viewing angle, visual and audio for correct orientation inside the VE. For example, scaled and viewing angle may be selected by empirical value based on experimental results (regression model and least square method - LSM). The other attributes (lighting and sound) are selected with expert's requirements and normative standards.

On the other hand, the environment should reproduce the imitation of basic tasks through reacting to the user's actions (movement, changed distance, time and physics laws). In this case, simple tasks (tracking and fitting) may be reduced in simple special tests (reaction on moving an object or changed object's distance).

The other things such as the physics laws or the movement may be corrected by developing tools (example Unity3D: colliders or rigid body). The main purpose is to create the immersion of a recipient in VE.

Then, after the process of immersion, there is a need to fill the environment with dynamic content. The high-level consists of building correct low-level and application logic based on the OCD. Therefore, the main target of this layer is to collect an unbound data in the complex training context based on a specific training simulator. There are many templates of OCD such as a complex 3D object or a mannequin.

Further research work should be focused on the low level of the design model. Especially, we will focus on scale and viewing angle based on experiments. A person will evaluate the distance between two points in the VE and real-world such as viewing angle. The results will be shown in the form of recommendation for the design of the training system, for example, simulator of harvesting machine.

 ISBN: 978-1-61208-750-4

ACKNOWLEDGMENTS

The results of this study were obtained with the support of Grant No. 25.1095.2017/4.6.

REFERENCES

[1] R. R. Burke, "Virtual Reality for Marketing Research," Innovative Research Methodologies in Management, Palgrave Macmillan, Cham, 2018, pp. 63-82.

[2] J. L.Rubio-Tamayo, B. M. Gertrudix and G. F. García, "Immersive Environments and Virtual Reality: Systematic Review and Advances in Communication, Interaction and Simulation," Multimodal Technologies and Interaction, 2017, vol. 1, № 4, 21.

[3] C. Boletsis, "The New Era of Virtual Reality Locomotion: A Systematic Literature Review of Techniques and a Proposed Typology," Multimodal Technologies and Interaction, 2017, vol. 1, № 4, 24.

[4] M. Cha et al., "A virtual reality based fire training simulator integrated with fire dynamics data," Fire Safety Journal, 2012, vol. 50, pp. 12-24.

[5] D. A. Bowman, J. L. Gabbard and D. Hix, "A survey of usability evaluation in virtual environments: classification and comparison of methods," Presence: Teleoperators & Virtual Environments, 2002, vol. 11, № 4, pp. 404-424.

[6] C. Boletsis, J. E. Cedergren and S. Kongsvik, "HCI research in Virtual Reality: A discussion of problem-solving," International Conference on Interfaces and Human Computer Interaction, IHCI 2017, Portugal, 21–23 July 2017, pp. 263-267.

[7] T. Gunn et al., "The use of virtual reality simulation to improve technical skill in the undergraduate medical imaging student," Interactive Learning Environments, 2017, pp. 1-8.

[8] T. Everson et al., "Astronaut training using virtual reality in a neutrally buoyant environment," DesTech 2017: Proceedings of the 2017 International Conference on Design and Technology, Knowledge E, 2017, pp. 319-327.

[9] S. Greenwald et al., "Technology and applications for collaborative learning in virtual reality," 2017.

[10] L. P. Berg and J. M. Vance, "Industry use of virtual reality in product design and manufacturing: a survey," Virtual reality, 2017, vol. 21, № 1, pp. 1-17.

[11] D. L. Neumann et al., "A systematic review of the application of interactive virtual reality to sport," Virtual Reality, 2017, pp. 1-16.

[12] E. Prasolova-Førland et al., Preparing for International Operations and Developing Scenarios for Inter-cultural Communication in a Cyberworld: A Norwegian Army Example," Transactions on Computational Science XXIII, Springer, Berlin, Heidelberg, 2014, pp. 118-138.

[13] M. Zyda, "From visual simulation to virtual reality to games," Computer, 2005, vol. 38, № 9, pp. 25-32.

[14] C. H. Lin and P. H. Hsu, "Integrating Procedural Modelling Process and Immersive VR Environment for Architectural Design Education," MATEC Web of Conferences, EDP Sciences, 2017, vol. 104, 03007.

[15] T. Everson and C. McDermott, "Astronaut training using Virtual Reality in a Neutrally Buoyant Environment," School of Engineering, Australia DesTech Conference Proceedings The International Conference on Design and Technology, 2017, pp. 319-326.

[16] M. Gonzalez-Franco and P. Rodrigo, "Immersive Mixed reality for Manufacturing Training: Frontiers in Robotics and AI," Frontiers in Robotics and AI, 2017, vol. 4, № 3.

[17] J. McComas, "CyberPsychology & Behavior," Children's Transfer of Spatial Learning from Virtual Reality to Real Environments, 1998, vol. 1, pp. 121-127.

[18] J. Broekens et al., "Virtual reality negotiation training increases negotiation knowledge and skill," International Conference on Intelligent Virtual Agents, Springer, Berlin, Heidelberg, 2012, pp. 218-230.

[19] J. Pirker, I. Lesjak and C. G. Maroon, "VR: A Room-Scale Physics Laboratory Experience," Advanced Learning Technologies (ICALT), 2017 IEEE 17th International Conference on IEEE, 2017, pp. 482-484.

[20] J. Rickel and W. L. Johnson, "Virtual humans for team training in virtual reality," Proceedings of the ninth international conference on artificial intelligence in education, 1999, vol. 578, p. 585.

[21] L. Schmid, A. Glässel and C. Schuster-Amft, "Therapists' Perspective on Virtual Reality Training in Patients after Stroke: A Qualitative Study Reporting Focus Group Results from Three Hospitals," Stroke research and treatment, 2016, vol. 2016, 12 p.

[22] I. Petukhov, L. Steshina and A. Glazyrin, "Application of virtual reality technologies in training of man-machine system operators," 2017 International Conference on Information Science and Communications Technologies, ICISCT 2017, 2017-December, pp. 1-7.

[23] G. Riva et al., "Affective interactions using virtual reality: the link between presence and emotions," CyberPsychology & Behavior, 2007, vol. 10, № 1, pp. 45-56.

[24] M. Slater et al., "Immersion, presence, and performance in virtual environments: An experiment with tri-dimensional chess," ACM virtual reality software and technology (VRST), New York, NY: ACM Press, 1996, vol. 163, 72 p.

[25] D. Bachmann, F. Weichert and G. Rinkenauer, "Review of three-dimensional human-computer interaction with focus on the leap motion controller," Sensors, 2018, vol. 18, № 7,. 2194.

[26] C. Repetto, P. Cipresso and G. Riva, "Virtual action and real action have different impacts on comprehension of concrete verbs," Frontiers in psychology, 2015, vol. 6, 176.

[27] H. G. Kim et al., "Measurement of exceptional motion in VR video contents for VR sickness assessment using deep convolutional autoencoder," Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology, ACM, 2017, p. 36.

[28] D. B. Kaber et al., "Investigating human performance in a virtual reality haptic simulator as influenced by fidelity and system latency," IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 2012, vol. 42, № 6, pp. 1562-1566.

[29] K. M. Stanney et al., "Human performance in immersive virtual environments: Effects of exposure duration, user control, and scene complexity," Human performance, 2002, vol. 15, № 4, pp. 339-366.

[30] J. Diemer et al., "The impact of perception and presence on emotional reactions: a review of research in virtual reality," Frontiers in psychology, 2015, vol. 6, p. 26.

[31] B. Herbelin, F. Vexo and D. Thalmann, "Sense of presence in virtual reality exposures therapy," Proceedings of the 1st International Workshop on Virtual Reality Rehabilitation, Lausanne, Switzerland, Citeseer, 2002.

[32] N. Parés and R. Parés, "Towards a model for a virtual reality experience: the virtual subjectiveness," Presence: Teleoperators and Virtual Environments, 2006, vol. 15, № 5, pp. 524-538.

[33] J. N. Latta and D. J. Oberg, "A conceptual virtual reality model," IEEE Computer Graphics and Applications, 1994, vol. 14, № 1, pp. 23-29.

 ISBN: 978-1-61208-750-4

[34] D. A. Bowman and R. P. McMahan, “Virtual reality: how much immersion is enough? ,” Computer, 2007, vol. 40, № 7.

[35] R. Pausch, D. Proffitt and G. Williams, “Quantifying immersion in virtual reality,” Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 1997, pp. 13-18.

[36] R. M. Satava and S. B. Jones, “Current and future applications of virtual reality for medicine,” Proceedings of the IEEE, 1998, vol. 86, № 3, pp. 484-489.

[37] J. B. Cooper and V. R. Taqueti, “A brief history of the development of mannequin simulators for clinical education and training,” Postgraduate medical journal, 2008, vol. 84, № 997, pp. 563-570.

[38] F. Biocca and M. R. Levy, “Communication in the age of virtual reality,” Routledge, 2013.

[39] A. Lele, “Virtual reality and its military utility //Journal of Ambient Intelligence and Humanized Computing,” 2013, vol. 4, № 1, pp. 17-26.

[40] V. Tanriverdi and R. J. K. Jacob, “VRID: a design model and methodology for developing virtual reality interfaces,” Proceedings of the ACM symposium on Virtual reality software and technology, ACM, 2001, pp. 175-182.

[41] J. P. Molina et al., “An interaction model for the TRES-D framework,” Electrotechnical Conference, 2006, MELECON 2006, IEEE Mediterranean, IEEE, 2006, pp. 457-461.

[42] C. H. Lin and P. H. Hsu, “Integrating Procedural Modelling Process and Immersive VR Environment for Architectural Design Education,” MATEC Web of Conferences, EDP Sciences, 2017, vol. 104, 03007.

[43] T. Cochrane et al., “A DBR framework for designing mobile virtual reality learning environments,” Australasian Journal of Educational Technology, 2017, vol. 33, № accepted for Special Issue on Mobile Augmented and Virtual Reality.

[44] W. Li, A. Y. C. Nee and S. K. Ong, “A State-of-the-Art Review of Augmented Reality in Engineering Analysis and Simulation,” Multimodal Technologies and Interaction, 2017, vol. 1, № 3, 17.

[45] A. M. Al-Ahmari et al., “Development of a virtual manufacturing assembly simulation system,” Advances in Mechanical Engineering, 2016, vol. 8, № 3, 1687814016639824.

[46] P. Kenny et al., “Building interactive virtual humans for training environments,” Proceedings of i/itsec, 2007, vol. 174, pp. 911-916.

[47] E. Prasolova-Førland et al., “Preparing for International Operations and Developing Scenarios for Inter-cultural Communication in a Cyberworld: A Norwegian Army Example,” Transactions on Computational Science XXIII, Springer, Berlin, Heidelberg, 2014, pp. 118-138.

[48] A. Prange et al., “A Multimodal Dialogue System for Medical Decision Support inside Virtual Reality,” Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, 2017, pp. 23-26.

[49] B. Ulicny and D. Thalmann, “Crowd simulation for interactive virtual environments and VR training systems,” Computer Animation and Simulation 2001, Springer, Vienna, 2001, pp. 163-170.

[50] H. K. Çakmak and U. Kühnapfel, “Animation and simulation techniques for vr-training systems in endoscopic surgery,” Computer Animation and Simulation, Springer, Vienna, 2000, pp. 173-185.

[51] F. Pürzel et al., “Real NC Control Unit and Virtual Machine to Improve Operator Training,” Procedia Computer Science, 2013, vol. 25, pp. 98-107.

[52] A. Nakai, Y. Kaihata and K. Suzuki, “The experience-based safety training system using VR technology for chemical plant,” Editorial Preface, 2014, vol. 5, № 11.

[53] T. Im et al., “A Virtual Reality based Engine Training System-A Prototype Development & Evaluation,” CSEDU, 2017, vol. 1, pp. 262-267.

[54] E. Stoll, M. Wilde and C. Pong, “Using virtual reality for human-assisted in-space robotic assembly,” Proc. World Congress on Engineering and Computer Science, 2009.

[55] X. Wang, P. S. Dunston and M. Skiniewski, “Mixed Reality technology applications in construction equipment operator training,” Proceedings of the 21st International Symposium on Automation and Robotics in Construction (ISARC 2004), 2004, pp. 21-25.

[56] K. M. Stanney, R. R. Mourant and R. S. Kennedy, Human factors issues in virtual environments: A review of the literature Presence, 1998, vol. 7, № 4, pp. 327-351.

[57] J. N. Latta and D. J. Oberg, “A conceptual virtual reality model,” IEEE Computer Graphics and Applications, 1994, vol. 14, № 1, pp. 23-29.

[58] M. J. Schuemie et al., “Research on presence in virtual reality: A survey,” CyberPsychology & Behavior, 2001, vol. 4, № 2, pp. 183-201.

[59] P. Gander, “Two myths about immersion in new storytelling media,” Lund University, 1999, Immersion and Emotion: Their Impact on the Sense of Presence.

[60] R. M. Baños et al., “Immersion and emotion: their impact on the sense of presence,” CyberPsychology & Behavior, 2004, vol. 7, № 6, pp. 734-741.

[61] I. Petukhov, L. Steshina and A. Glazyrin, “Application of virtual environments in training of ergatic system operators,” Journal of Applied Engineering Science, 2018, 16(3), pp. 398-403.

[62] S. Gupta et al., “Training in Virtual Environments: A Safe, Cost Effective, and Engaging Approach to Training,” 2008.

[63] M. J. Schuemie et al., “Research on presence in virtual reality: A survey,” CyberPsychology & Behavior, 2001, vol. 4, № 2, pp. 183-201.

[64] L. Piron et al., “Virtual Reality as an assessment tool,” Medicine Meets Virtual Reality 2001: Outer Space, Inner Space, Virtual Space, 2001, vol. 81, 386.

[65] C.Hendrix and W. Barfield, “Presence within virtual environments as a function of visual display parameters,” //Presence: Teleoperators & Virtual Environments, 1996, vol. 5, № 3, pp. 274-289.

[66] S. Sharples et al., “Virtual reality induced symptoms and effects (VRISE): Comparison of head mounted display (HMD), desktop and projection display systems,” Displays, 2008, vol. 29, № 2, pp. 58-69.

[67] H. Sveistrup, “Motor rehabilitation using virtual reality,” Journal of neuroengineering and rehabilitation, 2004, vol. 1, № 1, 10.

[68] F. P. Brooks, “What's real about virtual reality?,” IEEE Computer graphics and applications, 1999, vol. 19, № 6, pp. 16-27.

[69] G. Robertson, M. Czerwinski and M.Van Dantzich, “Immersion in desktop virtual reality,” Proceedings of the 10th annual ACM symposium on User interface software and technology, ACM, 1997, pp. 11-19.

[70] M. V. Sanchez-Vives and M. Slater, “From presence to consciousness through virtual reality,” Nature Reviews Neuroscience, 2005, vol. 6, № 4, 332.

[71] J. Jerald, “The VR book: Human-centered design for virtual reality,” Morgan & Claypool, 2015.

[72] Y. Liu, R. Feyen, and O. Tsimhoni, “Queuing Network-Model Human Processor (QN-MHP): A Computational Architecture for Multitask Performance in Human-Machine Systems,” ACM Transactionson Computer-Human Interaction, March 2006, vol. 13, № 1, pp. 37–70.

 ISBN: 978-1-61208-750-4

Mixed-Paradigm Framework for Model-Based Systems Engineering

Philipp Helle, Stefan Richter, Gerrit Schramm and Andreas Zindel
Airbus Central R&T
Hamburg, Germany
email: {philipp.helle, gerrit.schramm, stefan.richter, andreas.zindel}@airbus.com

***Abstract*—In a time when competition and market in aviation industry drive the need to shorten development cycles especially in early phases, both automation of processes and integration of tools become important. While constraints, such as make or buy decisions or corporate Information Technology (IT) governance influence the overall tool infrastructure in different directions, microservices are a fast-rising trend in software architecting. But that does not mean that the more traditional monolithic software architecture is dead. A resulting mixed-paradigm software applications can also be seen as an opportunity to profit from the best of both worlds. To support a newly developed complex system development approach called Smart Component Modeling, a supporting application framework prototype is subject to development with the objective to reduce both time and resources required during product development cycles. This paper describes the software architecture styles and deployment approaches that were used in a research project at Airbus for building a prototype and discusses challenges and opportunities that were encountered.**

Keywords–model-based systems engineering; microservices; REST.

I. INTRODUCTION

The MicroService Architecture (*MSA*) is a style that has been increasingly gaining popularity in the last few years [1] and has been called "one of the fastest-rising trends in the development of enterprise applications and enterprise application landscapes" [2]. Many organizations, such as Amazon, Netflix, and the Guardian, utilize *MSA* to develop their applications [2].

Pursuing the notion that "Microservices aren't, and never will be, the right solution in all cases" [3], this paper describes the architecture and development approach that was used in a research project at Airbus for building a prototype application framework for Model-Based Systems Engineering (*MBSE*). According to The International Council on Systems Engineering (INCOSE), "Model-Based Systems Engineering (*MBSE*) is the formalized application of modeling to support system requirements, design, analysis, verification and validation, beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [4]. This framework does not rely on a single paradigm but instead mixes different paradigms, viz. architecture patterns and deployment approaches, to achieve the overall goals: agility, flexibility and scalability during development and deployment of a complex enterprise application landscape.

This paper is structured as follows: Section II describes the modeling method that the built prototype *MBSE* framework is supposed to support. Section III provides background information regarding the different enterprise application architecture paradigms. Section IV explains the IT infrastructure in which the framework is deployed and Section V describes how and what features have been implemented in the prototype. Section VI discusses advantages and disadvantages of the mixed-paradigm approach. Section VII talks about the ongoing and future improvement effort before Section VIII wraps everything up with a conclusion.

II. SCM MODELLING METHOD

In [5], we provide a detailed account of the newly developed *MBSE* paradigm, called smart component modeling (*SCM*), that is rooted in a proposed change in the aircraft development process to include an out of cycle component development phase, in which components of an aircraft are developed independently of the traditional linear development process. These components are then adapted to the specific needs of a program within the more linear cycle. Furthermore, the paper describes a metamodel for modeling these so-called smart components based on proven *MBSE* principles [6]. Since the models are being defined outside of an aircraft program when requirements are not yet fixed, the models have to be parametric. An *SCM* is a self-contained model that can be developed out of cycle and enables capturing of all information relevant to the development of the component. *SCMs* are foreseen to be stored in a repository, called the *SCM Library*. This enables sharing and reuse. When the in-cycle phase of an aircraft or aircraft system development starts, the assets in the *SCM Library* are pulled and used as pre-defined and pre-verified components for a new development. The *SCM* metamodel defines all objects and their relations that are required to capture information related to smart components in models. The development of the *SCM* metamodel was driven by internal use cases and inspired by existing modeling languages such as the Systems Modeling Language (*SysML*) [7].

The requirements for the methodology supporting this new out of cycle process were as follows:

- The methodology shall be based on *MBSE* principles.
- The methodology shall be independent from any specific application domain.
- The methodology shall enable a product-line oriented product development, i.e., the metamodel must allow modeling of different variants of a product and ensure a consistent configuration and parametrization.
- The methodology shall enable inclusion of already existing domain models, i.e., models in a domain-specific modeling language.
- The methodology shall enable automatic verification of models, i.e., it shall be possible to check if the built models adhere to the modeling paradigm and to user-defined constraints.
- The methodology shall enable consistent modeling not only of the product itself but also of the context,

 ISBN: 978-1-61208-750-4

such as the industrial system used to built the product and allow the creation of relationships between the modeled artifacts.

The requirements for the application framework supporting this new modeling paradigm are as follows:

- The application framework shall be deployable in the current corporate IT infrastructure
- The application framework shall allow a heterogeneous technology stack to deliver the best solution for a designated purpose.
- The application framework shall be scalable with increasing number of models and users.
- The application framework shall be scalable in terms of model calculation performance.
- The application framework shall support continuous deployment strategies and agile frameworks to enable fast delivery and high flexibility.
- The application framework shall be efficient with regards to computing resources and reduce the companies ecological footprint.

III. Architecture paradigms

This section provides background information regarding the two main architecture paradigms that are used today: monolithic software and *MSA*. Service Oriented Architecture (*SOA*) and serverless architecture [8] are not described in detail as *SOA*, especially from a deployment perspective, still resembles monolith software [9] and serverless can be seen as taking *MSA* one step further [10].

A. Monolithic software

[11] defines a monolith as ”a software application whose modules cannot be executed independently”. This architecture is a traditional solution for building applications. A number of problems associated with monolithic applications can be identified:

- Due to their inherent complexity, they are hard to maintain and evolve. Inner dependencies make it hard to update parts of the application without disrupting other parts.
- The components are not independently executable and the application can only be deployed, started and stopped as a whole [12].
- They enforce a technology lock-in, as the same language and framework has to be used for the whole application.
- They prevent efficient scaling as popular and non-popular services of the application can only be scaled together [13].

Nevertheless, monolithic software is still widely used and, except for green-field new developments, there is hardly a way around it. [14] notes that a monolithic architecture is ”often a more practical and faster way to start”. Furthermore, if software from external parties is involved in a tool chain, it is not possible to change its architecture style.

B. Microservices

There is no single definition of what a *MSA* actually is. A commonly used definition by Lewis and Fowler says it is ”an approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API” [15]. Microservices typically consist of stateless, small, loosely coupled and isolated processes in a ”share-as-little-as-possible architecture pattern” [16] where data is ”decentralised and distributed between the constituent microservices” [17].

The term ”microservices” was first introduced in 2011 [15] and publications on architecting microservices are rapidly increasing since 2015 [18]. In 2016, a systematic mapping study found that ”no larger-scale empirical evaluations exist” [19] and concluded that *MSA* is still an immature concept.

The following main benefits can be attributed to *MSA*:

- Relatively small components are easier for a developer to understand and enable designing, developing, testing and releasing with great agility.
- Infrastructure automation allows to reduce the manual effort involved in building, deploying and operating microservices, thus enabling continuous delivery [18].
- It is less likely for an application to have a single point of failure because functionality is dispersed across multiple services [9].
- *MSA* does not require a long-term commitment to any single technology or stack.

[3] notes the obvious drawback of the current popularity of microservices that ”they’re more likely to be used in situations in which the costs far outweigh the benefits” even when monolithic architecture would be more appropriate.

In a study regarding the challenges of adopting microservices, [2] lists the distributed nature of *MSA*, which leads to debugging problems, the unavailability of skilled developers with intimate knowledge of *MSA* and finding an appropriate separation into services.

IV. Deployment Infrastructure

Corporate information technology (IT) environments imply very strict regularities when it comes to hard- and software architectures and deployments. Bringing in innovation in such an environment requires following a heterogeneous approach.

While it is more challenging to adapt hardware in a corporate context to cope with the latest innovations, service and software developments, e.g., ARM (Advanced RISC Machine) CPU (Central Processing Unit) platform based servers, GPU (Graphics Processing Unit) assisted computing or wide-usage of FPGAs (Field Programmable Gate Arrays), the application platform layer adaption is typically less demanding because almost any state-of-the-art deployment form, like bare-metal, Infrastructure-as-a-Service (*IaaS*), Platform-as-a-Service (*PaaS*) or Function-as-a-Service (*FaaS*) can be rolled out on standard server hardware.

The rationale for choosing a specific deployment form is based on various constraints imposed by corporate policies and long-term strategy decisions:

- Is the envisaged deployment form available in the corporate infrastructure?

 ISBN: 978-1-61208-750-4

- Has the deployment form limitations due to corporate policies, e.g., restricted internet access, restricted repository access?
- Are there any license limitations?
- Are there geolocation limitations for certain services, e.g., in a multinational company with multinational regulations according to law?
- Is the service available on premise or only on public cloud?
- Does a deployment form for a particular service fit in the long-term corporate IT strategy, e.g., make or buy decisions?

For the *Smart Component Modeling* prototype, it was necessary to make use of a heterogeneous software and hardware infrastructure provided by the corporate IT. Therefore, the deployment took place on *IaaS*, *PaaS* and *FaaS* platforms. Also, end user devices are involved, for example for running the *SCM workbench* (see Figure 2). That variety of platform types was chosen to provide inside information on how a new engineering concept could be supported by different software architecture approaches to be efficient in terms of development time, continuous integration (CI), resource efficiency and scalability.

A. Infrastructure-as-a-Service

In the context described above, *IaaS* is used to describe a hosting platform based on bare-metal and hosted hypervisors. It provides a variety of virtualized operating systems that are in compliance with corporate IT regulations.

For the prototype, the services hosted on classical virtual machines are mainly databases used as persistent layers for distributed Web applications. The main reason for not hosting the Web applications together with their respective persistence layer are resource restrictions. Current company policies prevent external access to the databases if they are part of the same microservice image as the hosting environment. This would either limit database management to a Web-based command line interface or require the implementation of a Web service deployed in the same container. Also, other external services could not be used to access the databases. This limitation is purely based on a decision made by the company's IT governance, but reflects day to day reality in corporate environments.

For any other Web application around the *SCM* prototype development, *IaaS* was avoided as the resource overhead cannot compete with *PaaS* or *FaaS*.

B. Platform-as-a-Service

In the following section, *PaaS* refers to an on-premise deployment of the *Red Hat Openshift*[20] platform. It is a platform built around *Docker*[21] containers orchestrated and managed by *Kubernetes* on a foundation of *Red Hat Enterprise Linux*.

In the prototype, *PaaS* plays a critical role for the continuous integration strategy. The image format used for the deployments follows the Source-to-image (*S2I*) concept. *S2I* is a toolkit and workflow for building reproducible container images from source code [22]. *S2I* produces ready-to-run images by injecting source code into a container image and letting the container prepare that source code for execution. The source code itself is hosted on an on-premise Github Enterprise[23] instance and the dependent resources are provided via an on-premise *Artifactory*[24] deployment that reflects the official sources of the required development environment such as Maven[25], npm, Python or NuGet.

The whole continuous deployment chain is secured via an exchange of keys and certificates to prevent disruptions for example due to company introduced password cycles for the developer and deployment accounts. The deployment speed is improved by using system instances for the *S2I* chain in the same geolocation of the company to prevent larger inter-site data transfers and round-trip times.

The microservice concept, together with *PaaS*, allows a massive reduction of resource allocations compared to an *IaaS* deployment, especially if the services are single and independent Web applications.

There are still limitations in the corporate environment that currently prevent larger scale use of the technology. The current setup allows a limited number pods per node, which becomes an issue when a service uses the scaling capability of the *OpenShift* platform. A second limitation is linked to the allocated sub-network and the deployment of the platform. All inter-service communication is routed via a unique company internal network. The *PaaS* instance does not re-use a network range that is already present in the company for inter-service communications as it would impose other challenges regarding communication from within the *PaaS* instance towards other company services. The rationale for the chosen *PaaS* implementation is primarily the reduction of classical virtual machines for simple hosting jobs and only secondary the creation of a massively scalable infrastructure for new service applications.

To cope with these limitations the prototype furthermore reduces the deployment footprint of single services for certain applications as described below.

C. Function-as-a-Service

FaaS is used for tiny stateless jobs, e.g., rendering of images. These services are monitored by an orchestrator that decommissions containers after idling for a defined time. This reduces resource usage further and has advantages in a scenario with a larger number of services.

The deployment architecture of the *FaaS* instance allows launching service containers within milliseconds. The applied software stack is *OpenFaaS* based on *Docker Swarm* running on a *Debian*[26] virtual machine.

One *FaaS* instance consumes resources similar to a pod on the above mentioned *PaaS* environment and hosts numerous services without performance limitations. While *PaaS* exposes containers under their distinct IP (Internet Protocol) addresses, *FaaS* comes with a reverse proxy that hides all containers and requires less IP addresses. This reduces the effort for routing name resolution and their documentation.

V. Implementation and Integration

The implementation of the prototype framework is split into different logical bricks as depicted by Figure 1. The *Architect Cockpit* allows a system architect to use existing models, to schedule the execution of simulations and to review results.

 ISBN: 978-1-61208-750-4

The *SCM Workbench* enables SCM developers to create and version *SCMs*. The *Back End* provides different services such as the orchestration of different processors to perform the execution of simulations.

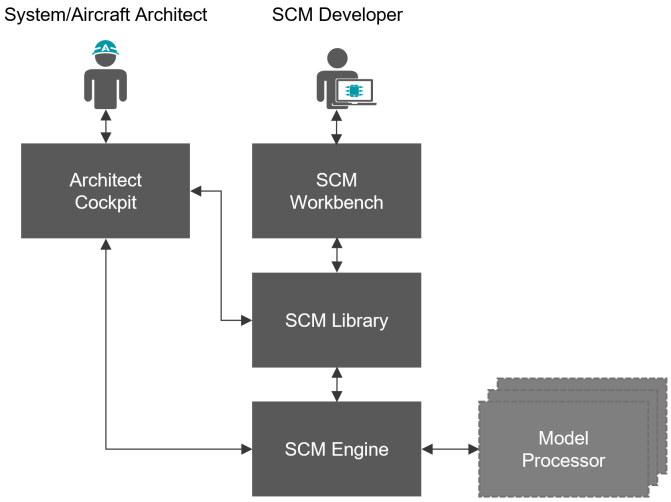

Figure 1. Prototype tool overview

A. Architect Cockpit

In order to reduce the workload and make the work for the architects as convenient as possible the interface for the cockpit is setup as an Angular single-site application. This allows using this entity without installing custom software and without bothering the user with update and migration procedures. The site is built using a Jenkins pipeline and then deployed on a specific git repository branch. A webhook on this branch triggers an *OpenShift* instance to build an Express.js server serving the previously build site on a *PaaS* cluster.

From a functional point of view the *Architect Cockpit* gives a reduced view on *SCMs*. Only information, which is necessary for the work of an architect is available and can be modified. This results in a nearly full intuitive usage of the interface and prevents faulty configurations. For example, some parameters can only be changed within a certain range. Ranges are defined by the model developer who knows the limitations best. The architect does not need to have a deep understanding of these limitations when using the predefined models.

B. SCM Workbench

The *SCM Workbench* is a full-fledged graphical editor to work with *SCMs* implemented as a monolithic rich-client application. It is implemented in an Eclipse Rich Client Platform *(RCP)* and based on the Eclipse Modeling Framework *(EMF)* [27]. It is a modeling framework and code generation facility for building tools and other applications based on a structured data model. *EMF* provides tools and run-time support to produce a set of Java classes from a model specification, along with a set of adapter classes that enable viewing and editing of the model, and a basic editor.

EMF is the basis for the Obeo Designer tool[28], which builds on the Eclipse Sirius project [29] and allows definition of graphic editors based on a defined *EMF* metamodel. This enables rapid prototyping of modeling solutions, which is ideal for a research/prototyping environment such as Airbus Central R&T. Changes to the metamodel are almost instantly available in the *SCM Workbench*, our prototype *SCM* modeling tool. On the other hand, *EMF* and *Obeo Designer* are mature and have been proven in industrial practice, e.g., *Capella*, the modeling tool from Thales that implements the *Arcadia* method is built with *EMF* and *Obeo Designer* as well [30].

Using such a rapid prototyping approach for the *SCM Workbench* can be easily misunderstood as just a proof-of-concept study. The final look and feel of the graphical editor for the *SCMs* is only limited by the amount of development time used for UX polishing. The workflow and information accessibility as well as the connection to a versioning system is comparable to other commercially available modeling tools, which are well known by the developers. It is assumed that a SCM developer has to take a short on-boarding training before using the *SCM Workbench*.

C. Back End

The *Back End* is build from several different entities that are based on different paradigms. These entities are described in the following paragraphs.

1) SCM Library: The *SCM Library* stores the models that have been created using the *SCM Workbench*. It is based on Connected Data Objects *(CDO)* a Java model repository for *EMF* models and metamodels. The specific implementation in use is the *Obeo Designer Team Server (ODTS)* which enables concurrent engineering of *EMF* models. A custom plug-in allows other services and applications to access the model repository through a *REST* interface. Due to its complex deployment strategy the *SCM Library* is deployed in an *IaaS* environment which allows more user interaction during updates.

2) SCM Engine: The *SCM Engine* can interpret *SCMs*, check constraints and run parametric calculations either as a single simulation run or as a *Design of Experiments* setup with multiple samples. It is a Java application executed in an OpenJDK Virtual Machine. Access to the engine is established through *REST* interfaces that are hosted on a Jetty server. The endpoints are described and documented using the Jersey framework. The *SCM Engine* is hosted on a *PaaS* instance and allows rolling updates, automated builds and scaling.

3) Model Processors: The *Performance Model API* serves as a glue between external domain-specific models with their own solver or simulation engine and the *SCM Engine*. A *Model Processor* is an application that implements this API to execute a specific model type. The API enables the *SCM Engine* to orchestrate simulations tools in a unified way and guides developers through the process of integrating additional simulation tools into this environment. In order to include a new model type in the *SCM* application framework, a model type specific *Model Processor* has to be implemented that implements the *Performance Model API* and connects to the model type specific solver or simulator. A reference implementation shows how this works for Excel models. An Excel model is processed by a Java application running in an OpenJDK VM using the Apache POI framework. Depending on the type of model and, e.g., the license and installation requirements of the model solver or simulator, the *Model Processor* can be deployed in any of the available deployment options *IaaS*, *PaaS* and *FaaS*.

 ISBN: 978-1-61208-750-4

Figure 2 depicts how the components of the *SCM* tool framework prototype are deployed in our infrastructure.

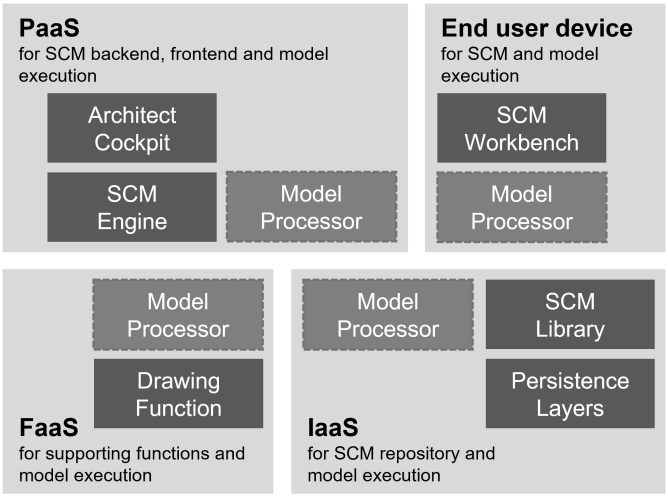

Figure 2. Prototype tool deployment

To make the polyglot approach of the *MSA* work and integrate each service all participating entities need to agree to a commonly understood interface. For the prototype *Representational State Transfer (REST)* over HTTP was chosen as the default interface combined with JavaScript Object Notation (*JSON*) as serialization format. *REST* over HTTP is a de facto standard since almost every technology stack provides at least an HTTP API if not specialized *REST* frameworks and clients such as *JAX-RS*. *JSON* as a serialization format is accepted and provides solid tooling on all integrated technologies. In addition many front-end frameworks natively support *JSON* such as *JavaScript* or *Ruby*. This eases the integration work needed to be done for the implementation of our demonstrators mainly the *Architect Cockpit*. As an added bonus it is easily digestible by human user, which helped tremendously with debugging. To built up process chains utilizing the deployed microservices we selected *Node-RED*. It provides all the tools necessary to handle *HTTP* based *REST* APIs and *JSON* based message bodies and is integrated well into the existing environment.

VI. Evaluation

Evaluating the mixed-paradigm approach, we experienced that developers where able create a working deployment much faster compared to the traditional approach using virtual machines. This also includes the amount of times that a new version of the service was built from once a week to several times a day using the automated CI pipeline. This increased the general development velocity as well as the prototypes feature set, which helped us to tailor the application to our stakeholder needs.

The raised deployment speed increased the number of times we experienced broken client applications. This was due to a violated interface contract between the services if the new features where not integrated properly. A well defined and adhered to interface specification is paramount for the success of introducing this mixed-paradigm approach.

In general, we noticed a greater sense of ownership of single developers over their service/code, which lead to a hike in the overall implementation quality. The mandatory usage of the git version control system increased the maintainability of the code base. The combination of git and the *Openshift* framework made it easy to recover from failures and faulty builds, which lead to a constant up-time of all services. In the future the introduction of additional agile software development principles like *Test Driven Development* could further increase the code quality.

The mixed-paradigm approach that was used to develop and deploy the prototype discussed in this paper led to reduced complexity, lower coupling, higher cohesion and a simplified integration. This in turn enabled agile collaboration for continuous delivery and integration of the solution.

VII. Outlook and future challenges

In the previous sections, we described how *MSA* can support the chosen polyglot approach utilizing a variety of different technology stacks and storage solutions. This enabled us to select the most fitting technical solution for the required functionality. Additionally the network based architecture provides an environment that is well suited for a multinational company like Airbus with sites scattered throughout different sites and IT domains. It also provided a commonly understood deployment layer for our cross-functional project team.

MSA supports us with the agility and velocity needed to convince our customers of our approach and implement a prototype that can handle the complexity of our *SCM* modeling approach. However, during the development we found stumbling blocks that need awareness once the scale changes from a research project prototype to a full scale industrial roll out.

Corporate IT – The proposed environment builds and hosts microservices in an agile and automated way. This requires the setup and maintenance of a CI pipeline (in our case *Openshift*/*GitHub*), which results in additional costs as well as an IT department that is capable of dealing with those investments. Additionally setting up certificate chains and firewalls to allow for secure communication inside the corporate network need to be accounted for. On the developer side roadblocks like proxy server hindering communication and enabling cross-origin resource sharing (*CORS*), which allows for communication between different domains need to be taken care of.

Service discovery – Once we reached a critical mass of microservices environment we discovered that it is hard to keep track of what services have already been implemented and what functionality each service provides. Even in our research project this point was reached rather quickly. Thus we introduced *Swagger*[31] as a Web based documentation for all our services and implemented a simple dashboard where services could be registered against. This allowed for manual service discovery across the team. In the future automated service discovery through bots and processable service descriptions will bring more value to the *MSA* approach by handling the sprawling service environment.

Now that we optimized the CI pipeline in the first half of the project we experience a rapid increase in deployed services. This allowed us to swiftly introduce new functionality as microservices, boosting the capabilities of our proof of concept prototype. It shows that *MSA* can initially speed up the

 ISBN: 978-1-61208-750-4

implementation velocity of a new project. Once we continue with the project more efforts will go towards managing the volume of services as well as (network) performance and reliability.

VIII. Conclusion

A direct, specific and measurable comparision between the described mixed-paradigm and a classical approach is not possible as it would have required the same infrastructure landscape to have been developed and deployed multiple times using different concepts. Nevertheless implementors were given the freedom to decide for every distinct artifact to freely choose the paradigm used for implementation. Furthermore developers were allowed to spilt artifacts which enables to select the right paradigm for each problem within. Later the interface documentation allowed the developers to easily re-implement an artifact using a different paradigm in case the initial decision for a specific paradigm reveals to have been not an optimal choice. Therefore the selection of the right paradigm appears to be inherent and native. To support a newly developed *MBSE* approach called Smart Component Modeling, a supporting application framework prototype had to be developed. Instead of a single architecture and deployment paradigm, a mixed-paradigm approach was followed to take the advantages of the different options and to consider external constraints coming from the IT governance. The software bricks were implemented in monolithic, *SOA*, microservice and serverless architecture glued together by *REST* interfaces over HTTP. The deployment took place on Desktop-PC, *IaaS*, *PaaS* and *FaaS* platforms. It provided insight into how a new engineering concept could be supported by different software architecture approaches to be efficient in terms of development time, continuous integration, resource efficiency and scalability.

References

[1] N. Dragoni et al., "Microservices: yesterday, today, and tomorrow," in Present and ulterior software engineering. Springer, 2017, pp. 195–216.

[2] J. Ghofrani and D. Lübke, "Challenges of microservices architecture: A survey on the state of the practice." in ZEUS, 2018, pp. 1–8.

[3] P. Jamshidi, C. Pahl, N. C. Mendonça, J. Lewis, and S. Tilkov, "Microservices: The journey so far and challenges ahead," IEEE Software, vol. 35, no. 3, 2018, pp. 24–35.

[4] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, Eds., Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th ed. Hoboken, NJ: Wiley, 2015.

[5] P. Helle, S. Feo-Arenis, A. Mitschke, and G. Schramm, "Smart component modeling for complex system development," in Proceedings of the 10th Complex Systems Design & Management (CSD&M) conference, forthcoming.

[6] A. Reichwein and C. Paredis, "Overview of architecture frameworks and modeling languages for model-based systems engineering," in Proc. ASME, 2011, pp. 1–9.

[7] Object Management Group, OMG Systems Modeling Language (OMG SysML), v1.2. OMG, Needham, MA, 2008.

[8] P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, "Serverless programming (function as a service)," in 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS). IEEE, 2017, pp. 2658–2659.

[9] A. Karmel, R. Chandramouli, and M. Iorga, "Nist definition of microservices, application containers and system virtual machines," National Institute of Standards and Technology, Tech. Rep., 2016.

[10] I. Baldini et al., "Serverless computing: Current trends and open problems," in Research Advances in Cloud Computing. Springer, 2017, pp. 1–20.

[11] N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, "Microservices: Migration of a mission critical system," arXiv preprint arXiv:1704.04173, 2017.

[12] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara, "From monolithic to microservices: an experience report from the banking domain," Ieee Software, vol. 35, no. 3, 2018, pp. 50–55.

[13] M. Villamizar et al., "Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud," in 2015 10th Computing Colombian Conference (10CCC). IEEE, 2015, pp. 583–590.

[14] ——, "Cost comparison of running web applications in the cloud using monolithic, microservice, and aws lambda architectures," Service Oriented Computing and Applications, vol. 11, no. 2, 2017, pp. 233–247.

[15] M. Fowler and J. Lewis. Microservices a definition of this new architectural term. [Online] http://martinfowler.com/articles/microservices.html [Accessed: 11 September 2019].

[16] T. Cerny, M. J. Donahoo, and M. Trnka, "Contextual understanding of microservice architecture: current and future directions," ACM SIGAPP Applied Computing Review, vol. 17, no. 4, 2018, pp. 29–45.

[17] D. Shadija, M. Rezai, and R. Hill, "Towards an understanding of microservices," in 2017 23rd International Conference on Automation and Computing (ICAC). IEEE, 2017, pp. 1–6.

[18] P. Di Francesco, I. Malavolta, and P. Lago, "Research on architecting microservices: trends, focus, and potential for industrial adoption," in 2017 IEEE International Conference on Software Architecture (ICSA). IEEE, 2017, pp. 21–30.

[19] C. Pahl and P. Jamshidi, "Microservices: A systematic mapping study." in CLOSER (1), 2016, pp. 137–146.

[20] RedHat, "Openshift," https://www.openshift.com/, 2019, [Online; accessed 21-October-2019].

[21] Docker Inc., "Docker," https://www.docker.com/, 2019, [Online; accessed 21-October-2019].

[22] A. Lossent, A. R. Peon, and A. Wagner, "PaaS for web applications with OpenShift origin," Journal of Physics: Conference Series, vol. 898, oct 2017, p. 082037.

[23] GitHub, Inc., "Github," https://github.com/, 2019, [Online; accessed 21-October-2019].

[24] JFrog Ltd, "Artifactory," https://jfrog.com/artifactory/, 2019, [Online; accessed 21-October-2019].

[25] The Apache Software Foundation, "Maven," https://maven.apache.org/, 2019, [Online; accessed 21-October-2019].

[26] Software in the Public Interest, Inc., "Debian," https://www.debian.org, 2019, [Online; accessed 21-October-2019].

[27] D. Steinberg, F. Budinsky, E. Merks, and M. Paternostro, EMF: eclipse modeling framework. Pearson Education, 2008.

[28] Obeo, "Obeo designer," https://www.obeo.fr/en/, 2019, [Online; accessed 21-October-2019].

[29] V. Viyović, M. Maksimović, and B. Perisić, "Sirius: A rapid development of dsm graphical editor," in IEEE 18th International Conference on Intelligent Engineering Systems INES 2014. IEEE, 2014, pp. 233–238.

[30] P. Roques, "MBSE with the ARCADIA Method and the Capella Tool," in 8th European Congress on Embedded Real Time Software and Systems (ERTS 2016), Toulouse, France, Jan. 2016.

[31] SmartBear Software, "Swagger," https://swagger.io/, 2019, [Online; accessed 21-October-2019].

 ISBN: 978-1-61208-750-4

JeroMF

A Software Development Framework for Building Distributed Applications Based on Microservices and JeroMQ

Aditi Jain
Computer Science Department
Utah State University
Logan, Utah, U.S.A.
email: aditi.jain@aggiemail.usu.edu

Stephen Clyde
Computer Science Department
Utah State University
Logan, Utah, U.S.A.
email: Stephen.Clyde@usu.edu

Abstract— **This paper describes the design, implementation, and testing of a software development framework, called *JeroMF*, that can help developers create scalable distributed applications based on a microservice architecture and that uses JeroMQ (a native Java implementation of ZeroMQ) for message passing. JeroMF includes an execution framework and extensible components for implementing processes, services, communication channels, messages, communication statistics, and encryption. Applications built with JeroMF do not require a message broker or any other middleware processes. However, they may include an optional Service Registry that can facilitate service discovery and secure communications. The Service Registry itself was implemented with JeroMF and is included as part of the JeroMF distribution. Thorough unit, integration, and system test cases exist for every component of JeroMF. For validation, JeroMF was used to re-design and re-implement a distributed health-care application with 13 separate types of services and very strict security requirements.**

Keywords-Microservices; Distributed Applications; Software Development Frameworks.

I. INTRODUCTION

Microservices are an architectural style for structuring applications around loosely coupled services and for making those services as granular as possible without compromising efficiency [1][2]. Microservices are highly maintainable, testable, independently deployable, and scalable [3]. Also, software engineers can organize them around business capabilities, thereby creating systems with excellent modularity and encapsulation, which can help with dynamic service composition and improve overall reliability, security, and fault tolerance. Microservices can also facilitate the continuous deployment of large, complex applications.

However, without a development framework, an application based on microservices can be hard to construct, test, debug, deploy, and maintain [3][4]. Simply splitting an application into multiple independent services generates more artifacts to manage without necessarily obtaining the desirable properties mentioned above. In fact, a haphazard refactoring of a distributed application into lots of independent services may create more complexity and thereby making maintenance and deployment more difficult.

When building an application based on microservices, developers need to modularize carefully, isolate relatively independent subsets of data together with the functionality for managing that data. Doing so will help reduce coupling and increase cohesion [5][6], and thereby improve reuse, maintainability, extensibility, and even scalability.

Also, when developers use microservices, they need to pay attention to all the typical implementation details for distributed applications, such as a) ensuring consistent implementation of communication protocols, b) ensuring the safety and consistency of transactions, c) achieving the desired amount of reliability despite communication or process failures, and d) guaranteeing the required level of security. Because a microservice-based application may have finer grain and diverse services and more communications than a similar application based on a client-server or service-oriented architectures, these challenges can be daunting and, if poorly handled, can cause the ultimate failure of the application.

This paper describes an open-source software development framework, called *JeroMF*, for creating distributed applications based on microservices efficiently and effectively. Specifically, JeroMF's goal is to make it easier for developers to create secure and reliable distributed applications by providing an execution framework and base components for processes, services, communication channels, messages, and communication statistics. JeroMF uses JeroMQ [7], a native Java port of ZeroMQ [8], as its communication library.

Section II provides some additional background on distributed applications in general, microservices, and JeroMQ. Then, Section III describes a sample application for illustration purposes. This is followed by an overview of JeroMF in Section IV. The full implementations for JeroMF and the sample application are available in public Git repositories. The URLs for these repositories are given later.

To verify JeroMF, we have created executable unit, integration, and system test cases. These test cases provide thorough test coverage using path and input domain partitioning testing techniques (see Section V). To validate JeroMF, we use it to re-design and re-implement a non-trivial

 ISBN: 978-1-61208-750-4

distributed application for the Utah Department of Health. This application, called the Child Health Advanced Records Management (CHARM) system. A brief summary of this case study is also provided in Section V. Finally, Section VI provides a summary and some thoughts about future work.

II. Background and Related Work

A. Distributed Applications

A distributed application is a software system that requires multiple processes to coordinate via network messages to complete its tasks. As such, they have to deal with both inter- and intra-process concurrency, as well as delays dues to message transfer [9]. Also, except for certain kinds of testing, the processes in a distributed application typically run on multiple independent hardware devices and therefore have to deal with the complexities of partial failure due to device or network failure [10]. Many mobile, Web-based, and enterprise applications today are actually distributed applications.

B. Microservices

To date, there is no concrete or widely accepted definition for microservices. Instead, microservices are general understood to be an architectural design concept, where the functionality of a distributed application is modularized into relatively small cohesive services. Each microservice works with its own data, can use other services, and can be implemented, tested, and deployed independent of other microservices [11].

Using microservices to build complex systems is not entirely a new idea. It stems from ideas central to Object-oriented Software Development [12] and that are found in many different types of architectures and design patterns, including Service-oriented Architecture [13], Domain-Driven Design [14], and Bounded Context [15]. Furthermore, they are consistent with software engineering principles, such as the Single Responsibility principle from SOLID [16] and the unified definitions of Abstraction, Modularization, and Encapsulation [17].

Some of the hoped-for benefits of microservices, include independent development, deployment, and scalability [4], as well as reusability, maintainability, and extensibility. Unfortunately, these benefits do not come for free. Developers must apply a wide range of expertise to address challenge inherit to distributed applications and to achieve designs with good modularity. Below is a summary list of some of these challenges identified in [4]:

- Increased complexity due to application features spanning multiple services;
- Increased complexity in setting up unit, integration, and system tests;
- The components or subpart of a real-world system often have poorly defined boundaries and, therefore, mapping them to services is non-trivial;
- Developers need to be expert in analyzing and balancing design decisions;
- Developers are responsible for the entire life cycle of a component (service);
- The complexities of state, when stateless services are not possible; and
- The complexities of communications, especially in achieving certain degrees of reliability and security.

C. Software Development Frameworks

In general, software development frameworks are collections of reusable components that provide execution infrastructures [18] and "inversion of control" [19]. With "inversion of control", developers don't have to write the main control logic directly and can focus on the functionality that is unique to an application [20], and can thus help developers to be more productive. Currently, there are many frameworks for developing distributed applications, such as Grails [21], Angular [22], and Coco [23] to name just a few. However, to our knowledge, none of them supports the creation of distributed applications using microservices and JeroMQ for communications.

D. ZeroMQ and Its Native Java Port, JeroMQ

In 2007, Pieter Hintjens along with Martin Sustrik introduced ZeroMQ as a high-performance, asynchronous, lightweight messaging library for scalable distributed applications [8]. ZeroMQ is fast, simple, and provides easy scalability. Also, it has been ported to over 40 programming languages, including a native implementation for Java, called JeroMQ [7]. Its application programming interface (API) for in-process, inter-process, peer-to-peer, and multicast communications is simple and consistent.

Developers working with ZeroMQ can create distributed application more quickly than with lower-level socket libraries because of its convenient abstractions and simple API. However, ZeroMQ is just a class library and not a development framework. As such, it does not directly provide an execution infrastructure or "inversion of control". Furthermore, it does not directly help developers with the challenges listed above.

III. Sample Application

To illustrate the architecture and use of JeroMF, we use a simple distributed application for managing used cars for multiple dealers (see Figure 1). With this sample application, every used-car dealer would run its own Used-car Server (only one shown in Figure 1) and each Used-car Server would contain a microservice, called Used-car Service. This service would encapsulate the dealer's own used-car data and provide a network-accessible API that would allow remote clients, e.g., the end user interface, to query what cars the dealer currently has in inventory and their prices.

 ISBN: 978-1-61208-750-4

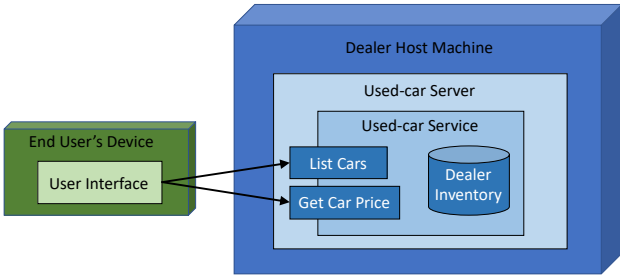

Figure 1. Sample Distributed Application for Tracking Used Cars

This sample application is minimal and only for illustration purposes. It does not contain all of the functionality one would expect in a real used-car application.

IV. OVERVIEW JEROMF

JeroMF is a framework that helps developers manage the complexities identified in Section II.B, so they can build quality distributed applications efficiently and effectively. Specifically, JeroMF aims to make it easy for developers to

1. Setup containers (processes) of services;
2. Manage service configuration parameters;
3. Create custom services that can a) access their own data stores, b) respond to incoming requests, and c) discover and use other services;
4. Define and implement reliable application-level communication protocols;
5. Use secure communications based on either asymmetric or symmetric encryption;
6. Monitor the status of all services in a distributed application;
7. Track service load and communication statistics;
8. Gracefully startup and shutdown services; and
9. Test services and inter-service communications.

A. Architectural Overview

The Unified Modeling Language (UML) Class Diagram [12][24] in Figure 2 shows JeroMF's primary packages with their essential classes and relations. From left to right are the base components for implementing custom processes, application-specific services; and communications. Developers create distributed applications in JeroMF by implementing specializations of these components or by reusing them directly. The following sections describe them in more detail, beginning with the process-related components.

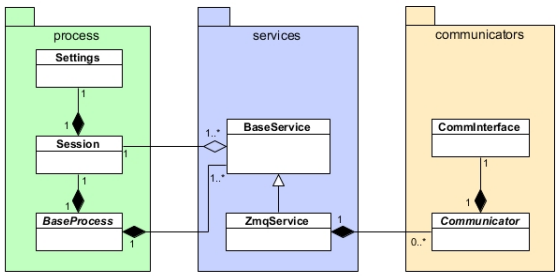

Figure 2. The primary packages in JeroMF with their key classes and relationships.

B. Processes

A process in JeroMF, defined as a specialization of BaseProcess, is an execution container that holds one or more services. If a developer is following a strict microservice architecture, then each JeroMF process will hold exactly one service. However, JeroMF allows a process to hold more than one service, at the developer's discretion, to achieve better execution and deployment efficiencies in certain situations.

A JeroMF process also contains a Session object, which in turn contains a Settings object. The Session object keeps track of the process's name, status, Settings object, JeroMQ context, and encryption keys. The Settings object holds all the configurable settings for a process and its services. Each setting has value that can be changed at runtime through either property files, environment variables, or command-line parameters. The Session object is shared with all the process's services so they can make use of that information.

Figure 3 contains a Class Diagram of used-car application, with the components implemented by the developer in light

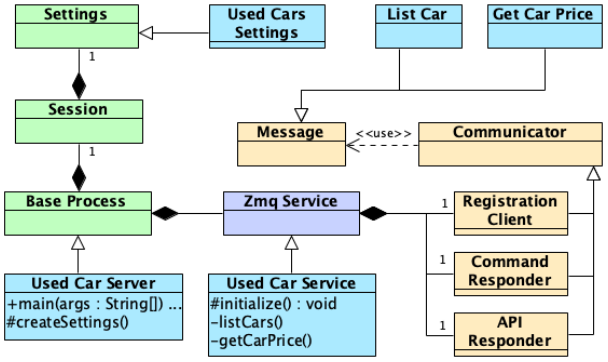

Figure 3. Classes in the Used-car Application, with those implemented by developer shown in light blue.

```
public class UsedCarServer extends BaseProcess {
  public static void main(String[] args){
    UsedCarServer process=new UsedCarServer();
    try {
      process.initialize(args,"server.config");
      UsedCarService service = new
          UsedCarService(instance.getSession(),
          "UsedCarsService");
      process.addService(service);
      process.run();
    }
    catch (Exception e) {e.printStackTrace(); }
    finally { process.cleanup(); }
  }

  @Override
  protected Settings createSettings() {
    return new UsedCarSettings();
  }
}
```

Figure 4. Implementation of the UsedCarServer class

 ISBN: 978-1-61208-750-4

blue. Figure 4 contains the code for UsedCarProcess from the sample application and is typical of most JeroMF processes. When a process starts, the main() routine calls the initialize() method – a Template Method [25] that setups the Session and Settings objects using virtual methods that the developer can override in the specialization. For example, the UsedCarProcess needs a custom Settings, so developer simply implements a specialization of Settings (not shown) and then overrides createSettings() method to return an instance of that specialization. See Figure 3.

After the process is initialized, the main() method instantiates the service that it will contain, adds it to the process, and then calls a run() method to begin execution. The run() method will only return once the process is stopped, which typically occurs after a service receives a Shutdown command or when it determines that the process needs to enter a terminal state. Finally, once the run() method does return, main() method will call cleanup().

C. Services

The BaseService class (see Figure 2) represents a basic microservice with an optional database connection. It has access to the process's Session object, which is provided as a parameter to the constructor. The ZmqService class is a specialization of BaseService class that represents a microservice with communication capabilities based on JeroMQ. As such, it can have zero or more communicators, i.e., instances of the Communicator class, for interacting with clients or other services. Typically, and by default, a ZmqService would include three communicators:

- a registration client that is responsible for listing the service with the Registry (if application uses a Registry), so other processes can find it and for setting up secret keys for symmetric encryption,
- a command responder that listens for general control messages from the Registry or some other control process, and
- an API responder for handling requests from clients.

None of these communicators are required and are only setup if their configuration settings have values in the Settings object.

Although BaseService and ZmqService can be used as-is for instantiating many types of services, they can be further customized through specialization. Like JeroMF processes, services have initialize() and run() methods that follow the Template Method pattern, with the customizable parts encapsulated in virtual methods.

Figure 5 contains a specialization of ZmqService, called UsedCarService, for the used-car application. When a UsedCarService is initialized, which happens when the service is started, it calls its super's (i.e., ZmqService's) initialize() method, which automatically sets up instances of the three types of communicators listed above.

```
public class UsedCarService
        extends ZmqService {

 UsedCarService(Session session, String srvName)
       throws ServiceException {
   super(session, srvName);
 }

 @Override
 protected void initialize()
        throws ServiceException {
  super.initialize();
  apiResponder.addMessageHandler(ListCars.class,
     EncryptionMode.None,
     EncryptionMode.None,
     msg -> listCars());
  apiResponder.addMessageHandler(GetCarPrice.class,
     EncryptionMode.None,
     EncryptionMode.None,
     msg -> getCarPrice(msg));
 }

 private Message listCars(){ … }

 private Message getCarPrice(Message request){ … }
}
```

Figure 5. Code snippet of UsedCarService

ZmqService's initialize() method also calls its super's (i.e., BaseService's) initialize() method, which sets up everything that is needed for working with the database. The actual opening the database connection is deferred until the first time it is used, thereby minimizing initialization time

After calling its super's initialize() method, UsedCarService's initialize() method customizes its API Responder to handle two types of messages, namely ListCars and GetCarPrice, by setting up message handlers for them. A message handler for a type of message defines what kind of encryption to expect for the incoming message and what type of encryption to use for the reply, along with a lambda function for processing incoming messages. In this example, the both lambda function simply call private methods. The private methods (implementations not shown) get a reference to database connection using a protected method inherited from BaseService and then use that connection to retrieve the requested information. They return a reply message or a null, if the desired information could not be retrieved.

D. Communicators

Communicator is an abstract base class for the objects that handle all the communications in JeroMF. A communicator uses JeroMQ, which in turn uses one of three transport-layer communication mechanisms, namely: *Transmission Control Protocol* (TCP), in-process (Inproc), or inter-process communication (IPC) [26]. Each communicator has an end point that defines both the transport-layer communication mechanism and either the local address that the communicator will bind to or the remote end point that it will connect to. The details about a communicator's end point are encapsulated in an instance of CommInterface class. Developers do not need to directly create or access these objects.

JeroMF includes six reusable communicators:

 ISBN: 978-1-61208-750-4

- The *Requester* and *Responder* communicators handle reliable request-reply style communications where the requester initiates all conversations
- The *Active Responder* and *Passive Requester*, which also handle reliable request-reply style communications, but the responder starts by indicating its readiness to receive requests
- The *Command Publisher* and *Command Responder*, which provide for simple but secure one-way message broadcasts.

JeroMF also includes a special type for Requester, called *RegistrationClient*, that registers services with the optional Registry process. This was mentioned above as one of the standard communicators for a ZmqService.

All communicators can send and receive encrypted or unencrypted messages. For encrypted messages, a communicator may use either asymmetric encryption based on a public-private key pair or symmetric encryption based on a shared secret key. For asymmetric encryption where a communicator needs to encrypt or decrypt with a private key, a ZmqService will give the communicator the name of the key pair and the password for opening the private key. It should get these values from the Settings object. For asymmetric encryption where a communicator needs to decrypt or encrypt a message with a public key, it can ask its ZmqService to lookup the public key by name. If the distributed application is using a Registry, then a ZmqService can use the Registry to discover this public key, if it is not already known.

Since communicators send and receive messages, JeroMF provides a base class, called Message, for implementing message structures quickly. Developers simply have to create specializations of this base class and then define appropriate data members with getters and setters.

V. TESTING AND EVALUATION

A. Verification

JeroMF was tested at the unit, integration, and system level with executable test cases using JUnit [27]. For unit testing, we used a combination of path testing [28] and input domain partitioning testing [29] techniques and achieved reasonably good coverage by striving to meet the following criteria:

- Every statement is executed in at least one test case.
- Every possible outcome of each conditional clause is tested in at least one test case.
- Representative examples of each boundary case for every looping construct is executed in at least one test case.
- Every possible exception is thrown in at least one test case.
- Representative examples from each partition element of each input domain for each method is used in at least one test case.

During the unit testing, we discovered that some of the declared exceptions from JeroMQ and other 3rd party libraries are impossible to stimulate in automated test cases. So, our coverage for unit testing is not 100%, but it is very close.

For integration and system testing, we also created executable unit test cases using Junit. However, each of these test cases have to ensure that other services are running and, if not, start them up before executing and shut them down afterwards. To this end, we created some utility components for checking the status of another service, for launching a process that contains that service, and for eventually shutting that process down. These utilities components allow us to create automated integration and system test cases, giving us confidence that the individual components of JeroMF are working together correctly and that the framework as a whole is satisfying its requirements.

B. Validation

Validating JeroMF requires using it to develop real distributed applications. Over the last 20 years, Utah State University has developed a number of distributed applications for the Utah Department of Health, including an information broker, called the *Child Health Advanced Record Management* (CHARM) system [30]. This system allows health-care professionals to view a wide range of health-care data for a given child from multiple data sources, securely and in real-time. To do its job, CHARM must monitor and interact with multiple data sources and data consumers, matcher child records across the data sources, identify special situations about which health-care professionals need to be alerted, and monitor itself.

This distributed application, which has been operating since 2006, seemed like a good candidate to re-design and re-implement using JeroMF. It is complex, requires high levels of security, maintainability, and extensibility. So, as an initial case study, we selected a major portion of this system, called the Sync Facility, and re-built this subsystem using JeroMF.

After refactoring into microservices, the Sync Facility ended up with 16 different types of services, hosted in 13 processes. The refactoring simplified the architectural design of the Sync Facility and improved its ability to be tested and deployed. Though antidotal evidence, the developers also believe that the new Sync Facility will be more maintainable and extensible.

C. Continuous Integration and Deployment

All of JeroMF (i.e., its base components, Registry, and utilities) and the used-car example are contained in the public Git repositories on Bitbucket.org, under the "usucssedevelopment" user [31]–[34]. Specifically, the base repository [31] contains the JeroMF source code and test cases. It compiles to a distribution package that distribution

 ISBN: 978-1-61208-750-4

application will import to use JeroMF. It is configured to use CircleCI [35] for continuous integration and to automatically deploy its distribution package to a Maven repository. The second repository [32] contains the Registry and is itself a program built with JeroMF. The third repository [33] contains some utility components, such as a process launcher, that are used for the integration and system testing of JeroMF but can also help with the deployment and launching of distributed applications, in general. The fourth repository contains the a barebones but functional implementation of used-car example [34].

VI. CONCLUSION

Our initial experience with JeroMF has provided preliminary evidence that it is valuable framework for implementing distributed applications based on microservices and JeroMQ. Its BaseProcess class makes it easy to define new service containers that can run on bare-bones Java platforms, i.e., a platform with no Web servers or application servers. Its BaseService and ZmqService classes make it easy to create custom microservices that can implement diverse and sophisticated functionality. The predefined Communicator and Message classes allow developers to implement common styles of communication and provide excellent starting points for implementing application-specific communication protocols. Also, the Communicator class makes it easy for developers to use either asymmetric or symmetric encryption. Furthermore, the optional Registry process can act like a key store for the public keys of registered services, simplifying key management.

The JeroMF services also have built in monitoring logic that can allow monitoring processes to either actively query the service status or receive periodic updates from services. Services can also track statistics about workloads and message traffic, and then provide that information to monitoring processes for analysis. Finally, the standard Command Responder for a service provides a simple but secure way to shut down or restart services.

Despite its rich set of features, JeroMF is still in its infancy. We envision several important enhancements to JeroMF in the near future. First, we aim to create other specializations of BaseService, like ZmqService, that would support different messaging libraries. For example, we plan to create an HttpService that uses HTTP [36] instead of JeroMQ and that has built-in support for RESTful [37] operations. After that, we plan on implementing and testing extensible services that will act as request proxies and load balancers.

We also plan to conduct several empirical studies and qualitative analyses that will aim to answer questions about its utility, reusability, extensibility, scalability, security, reliability, and maintainability. In preparation for some of these studies, we will track detailed information about software problem reports, time to resolution, induced errors from bug fixes, and more.

Finally, we plan to create more public examples that can help explain how to use JeroMF in build production-quality distribution applications and to serve as testbeds for empirical studies.

We welcome feedback and contributions from developers who would like to use JeroMF to build distributed applications.

REFERENCES

[1] "Microservices," *martinfowler.com*. Available from: https:// martinfowler.com/articles/microservices.html. [retrieved: Sept., 2019].

[2] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural Patterns for Microservices: A Systematic Mapping Study.," presented at the CLOSER, 2018, pp. 221–232.

[3] P. Hauer, "Microservices in a Nutshell. Pros and Cons.," *Phillip Hauer's Blog*. Available from https://phauer.com/ 2015/microservices-nutshell-pros-cons/. [retrieved: Sept., 2019].

[4] D. Kerr, "The Death of Microservice Madness in 2018," *Dave Kerr's Blog*, 12-Jan-2018. Available from: https:// dwmkerr. com/the-death-of-microservice-madness-in-2018/. [retrieved: Sept., 2019].

[5] G. Gui and P. D. Scott, "Coupling and Cohesion Measures for Evaluation of Component Reusability," in *Proceedings of the 2006 International Workshop on Mining Software Repositories*, New York, NY, USA, 2006, pp. 18–21.

[6] I. Candela, G. Bavota, B. Russo, and R. Oliveto, "Using Cohesion and Coupling for Software Remodularization: Is It Enough?," *ACM Trans Softw Eng Methodol*, vol. 25, no. 3, pp. 24:1–24:28, Jun. 2016.

[7] *Pure Java ZeroMQ. Contribute to zeromq/jeromq development by creating an account on GitHub*. The ZeroMQ project, 2019.

[8] "ZeroMQ." Available from: https://en.wikipedia.org/wiki/ ZeroMQ. [retrieved: Sept., 2019]

[9] G. Coulouris, J. Dollimore, T. Kindberg, and G. Blair, *Distributed Systems: Concepts and Design*, 5 edition. Boston: Pearson, 2011.

[10] J. Link, "Chapter 11 - Distributed Applications," in *Unit Testing in Java*, J. Link, Ed. San Francisco: Morgan Kaufmann, 2003, pp. 225–240.

[11] E. Wolff, *Microservices: Flexible Software Architecture*, 1 edition. Addison-Wesley Professional, 2016.

[12] G. Booch et al., *Object-Oriented Analysis and Design with Applications*, 3 edition. Upper Saddle River, NJ: Addison-Wesley Professional, 2007.

[13] "What Is SOA?," Available from: https://web.archive.org/ web/20160819141303/. [retrieved: Aug., 2019].

[14] E. Evans, *Domain-Driven Design: Tackling Complexity in the Heart of Software*, 1 edition. Boston: Addison-Wesley Professional, 2003.

[15] M. Fowler, "BoundedContext," *martinfowler.com*, Available from: https://martinfowler.com/bliki/BoundedContext.html. [retrieved: Sept., 2019].

[16] S. Metz, "SOLID Object-Oriented Design - GORUCO 2009." Available from: http:// https://www.youtube.com/watch?v=v-2yFMzxqwU. [retrieved: Sept., 2019].

[17] S. Clyde and J. E. Lascano, "Unifying Definitions for Modularity, Abstraction, and Encapsulation as a Step Toward Foundational Multi-Paradigm Software Engineering Principles," in *Proceedings of the Twelfth International*

Conference on Software Engineering Advances, Athens, Greece, 2017.

[18] "Library vs. Framework?," *Program Creek*, Available from: https://www.programcreek.com/2011/09/what-is-the-difference-between-a-java-library-and-a-framework/. [retrieved: Sept., 2019].

[19] "Inversion of Control Containers and the Dependency Injection pattern," *martinfowler.com*. Available from: https://martinfowler.com/articles/injection.html. [retrieved: Sept., 2019].

[20] "The Difference Between a Framework and a Library," *Developer News*, 01-Feb-2019. Available from: https://www.freecodecamp.org/news/the-difference-between-a-framework-and-a-library-bd133054023f/. [retrieved: Sept., 2019].

[21] "Grails Framework." Available from: https://grails.org/. [retrieved: Sept., 2019].

[22] "Angular." Available from: https://angular.io/. [retrieved: Sept., 2019].

[23] "Coco: A New Open-Source Blockchain Framework – MontageJS." *MontageJS*. Available from: http://montagejs.org/coco-open-source-blockchain. [retrieved: Sept., 2019].

[24] A. S. Evans, "Reasoning with UML class diagrams," in *Proceedings. 2nd IEEE Workshop on Industrial Strength Formal Specification Techniques*, 1998, pp. 102–113.

[25] E. Gamma, R. Helm, R. Johnson, J. Vlissides, and G. Booch, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1 edition. Reading, Mass: Addison-Wesley Professional, 1994.

[26] P. Hintjens, *ZeroMQ: messaging for many applications*. O'Reilly Media, Inc., 2013.

[27] "JUnit – About." Available from: https://junit.org/junit4/. [retrieved: Sept., 2019].

[28] A. Watson and T. McCabe, "Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric," National Institute of Standards, NIST Special Publication 500-235, Sep. 1996.

[29] J. Tian, "Input Domain Partitioning and Boundary Testing," in *Software Quality Engineering: Testing, Quality Assurance, and Quantifiable Improvement*, IEEE, 2005, pp. 127–146.

[30] S. Clyde, *Child-Health Advanced Record Management Systems*. Salt Lake City, Utah, USA: Utah Department of Health, 2006.

[31] S. Clyde and A. Jain, *JeroMF Base Components*. Logan, Utah, USA: Utah State University, 2019. Available from: https://bitbucket.org/usucssedevelopment/base.git. [retrieved: Sept., 2019].

[32] S. Clyde and A. Jain, *JeroMF Registry*. Logan, Utah, USA: Utah State University, 2019. https://bitbucket.org/usucssedevelopment/registry.git. [retrieved: Sept., 2019].

[33] S. Clyde and A. Jain, *JeroMF Utilities*. Logan, Utah, USA: Utah State University, 2019. https://bitbucket.org/usucssedevelopment/utils.git [retrieved: Sept., 2019].

[34] S. Clyde and A. Jain, *JeroMF Used-car Example*. Logan, Utah, USA: Utah State University, 2019. https://bitbucket.org/usucssedevelopment/jeromfexamples-usedcars.git [retrieved: Sept., 2019].

[35] "Continuous Integration and Delivery," *CircleCI*. Available form: https://circleci.com/. [retrieved: Oct., 2019].

[36] J. F. Reschke and R. T. Fielding, "Hypertext Transfer Protocol (HTTP/1.1): Authentication." Available from: https://tools.ietf.org/html/rfc7235. [retrieved: Sept., 2019].

[37] "What is RESTful API? - Definition from WhatIs.com," *SearchMicroservices*. Available from: https://searchmicroservices.techtarget.com/definition/RESTful-API. [retrieved: Sept., 2019].

 ISBN: 978-1-61208-750-4

Automata-Based Timed Event Program Comprehension for Real-Time Systems

Aziz Fellah and Ajay Bandi

School of Computer Science and
Information Systems
Northwest Missouri State Univerity
Maryville, Missouri USA
Email:{afellah, ajay}@nwmissouri.edu

***Abstract*—In this paper, we extend the software space of program comprehension to real-time systems and introduce two orthogonal and hybrid paradigms that we refer to as timed event component comprehension and timed event program comprehension. The former, timed event component comprehension, with no role in the coding aspect, is a set of autonomous timed event components that provide a high-level system-specific functionality about the overall real-time system including its structure, components and their synchronized interrelationships at different level of granularity, and static and dynamic behaviors. The later, timed event program comprehension recovers high level of information from timed event component comprehension and then builds an automata-based model about the system. This process occurs before carrying any program comprehension. We show that both paradigms are intrinsically linked and neither of them can be explored in isolation. Importantly, we map the component comprehension paradigm into a distinguished component class that we refer to as timed event components (TeCmp) which, in turn, are formally modeled as timed event automata, a powerful canonical model for modeling and verifying real-time computations. Furthermore, to support this research towards an effective program comprehension geared towards real-time and embedded systems, we investigated and evaluated the effect of our approach through a practical Internet of Things (IoT) case study.**

Keywords–Program comprehension; program understanding; software modeling; real-time systems; embedded systems; IoT; timed event automata.

I. INTRODUCTION

Because of its importance in software engineering, program comprehension has emerged as a significant component in software evolution and maintenance. It is a process of understanding an existing software system before it can be properly maintained, enhanced, reused, and extended. For instance, a common situation that software developers may find themselves in is reviewing and extending their own or their teammate's code. This situation is much easier than understanding and maintaining the code of unfamiliar software systems, or reading the code of an Application Programming Interface (API)/utility library. We call these knowledge-intensive activities program comprehension, which is considered as an important aspect of the software development process. In general, new developers spend much of their time analyzing code and searching for information to understand the system under evolution. Other closely related terms are also used to describe activities related to program comprehension, such as code refactoring and reverse engineering. For years, researchers have tried to understand how developers comprehend programs during software maintenance and evolution, and assess the quality of program comprehension. To address these challenges, numerous proposals and approaches have been investigated by Storey [1], Siegmund [2], Yuan *et al.* [3], Fowkes *et al.* [4], and Lucia *et al.* [5], just to name a few that span a spectrum of activities, such as cognitive models and software visualization, empirical evaluation, mental models representation of the program, knowledge-base models, top-down and bottom-up comprehension, code semantics, and data context interaction [1], [6]–[8]. Some of these theoretical models are grounded in experimental studies and validated by experienced programmers.

In this paper and with no comprehensive overview, we attempt to lay a foundation of program comprehension for real-time systems, an area of research that has not received much attention and could be investigated in various directions. In this work, we are not claiming that we developed a general and conclusive program comprehension framework for real-time and embedded systems, but our work will add value to the existing approaches. The paper describes strategies and knowledge needed as well as the rational of this orthogonal paradigm: component and program comprehension. We will shed light on what developers should emphasize when faced with the challenging time-dimension tasks of gaining an understanding of real-time source code. This should be aligned with the original code of the designers.

Importantly, the focus of this contribution is on two orthogonal and hybrid paradigms that we introduce and refer to as *timed event component comprehension* and *timed event program comprehension*. Such a dual comprehension paradigm would help programmers with comprehending systems' functionality, understanding code, interweaving abstractions, and building a mental model about a piece of software as well as using effective tools to support program comprehension activities.

Timed event component comprehension provides a high level system-specific functionality about the overall real-time system including its architectural structure, static and dynamic behaviors, and synchronized interrelationships at different levels of granularity. With no role in the coding aspect, timed event component comprehension tasks are grounded on a set of autonomous functional block units that we refer to as *timed*

 ISBN: 978-1-61208-750-4

event components (TeCmp). The abbreviation of TeCmp will be used for both singular and plural terms in the concordant context of the sentence in the rest of this paper.

Timed event program comprehension recovers high-level of information from timed event component comprehension and builds an automata-based model about the system before carrying program (*i.e.,* source code) comprehension. The coordination and interaction between TeCmp is fully delegated to a special class of components that we refer to as *timed component connector* (TeCnn).

A major challenge in the proposed timed event component comprehension development is the coordination of the active components and entities that comprise real-time systems. Thus, there is a need to complement TeCmp with formalisms for coordinating, integrating, and synchronizing components which have well-defined and fixed interfaces. In addition, we collectively refer to the pair, *timed event component* and *timed event connector* models, as (TeC&C) which are formally modeled by timed event finite automata, a powerful canonical model for modeling and verifying real-time computations.

The structure of the paper is as follows. In Section II, we survey the related work and research challenges that appear in software systems related to program comprehension. In Section III, we describe timed event component-based framework which is characterized in terms of two types of components that we refer to as TeCmp and TeCnn. Both of these components are intrinsically linked and neither of them can be explored in isolation. Section IV discusses the challenges of component comprehension in real-time systems. Furthermore, this section states some definitions and concepts that can be used in subsequent sections. Section V focuses on timed event component and connector models (TeC&C) to gain an understanding of the overall system‘s inner workings in terms of the time dimension. Section VI describes time event transitions which are fundamentally important for real-time systems. It also maps the main component, such as TeCmp into timed event automata. The characteristics of the IoT irrigation case study system are presented and summarized in Section VII. We conclude the paper with some potential discussions in Section VIII.

II. Related Work

Over decades program comprehension has been characterized by several classical theories and strategies in conjunction with other complementary techniques such as software inspection, visualization [9], static and dynamic source code analysis. For instance, the knowledge-base model of [10] which is based on the problem domain, developer's experience and background knowledge. A number of mental representations at various levels of abstraction have been investigated in literature [1]–[6] [11]. The top-down model [1] which reflects the developer's mental and conceptual representations integrate domain knowledge as a starting point. On the contrary, with no prior knowledge and little experience with the domain, program comprehension starts at the source code level and builds a higher-level abstraction (bottom-up model) [11]. Knowledge-based, mental and top-down models support the timed event component comprehension paradigm. However, the bottom-up model supports timed event program comprehension paradigm. Based on the nature of events, time-driven and event-driven of [12] and [13], real-time UML (Unified Modeling Languages) has emerged as the choice of the development of real-time and embedded systems. Data context interaction architecture [8] is a software paradigm whose main goal is to bring the end user's mental models and computer program models closer. Data context interaction [8] focuses on objects and their relationships to mental models by which users and programmers add new functionalities and modify the existing ones.

Furthermore, the software system development has shifted its emphasis from traditional building and programming software systems to a component-based approach. Component-Based Development (CBD) [14]–[18] has emerged among the most feasible approaches to overcome and address the software complexity in different domain areas, and advocates the reuse of independently developed software components as a promising technique for the development of complex software systems. Importantly, individual component-based functionalities incorporate potential future reusability, hence served to increase the program comprehension.

Our approach is different from other existing conceptual and theoretical models because we are primarily focusing on the timing characteristics of the application, which is the most predominant factor in real-time and embedded systems. In general, our work partially borrows the concept of time stamps of Leslie Lamport [19], but in particular it is grounded on the foundation of timed automata of Alur [12].

III. Timed Event Component-Based Development

The component-based model [14]–[16] is used to develop software at higher abstraction levels and promotes the reuse and evolution of existing artifacts and entities developing new software systems. It is composed of a collection of functional building blocks or services that have become a system blueprint in modern software engineering development life cycle. In timed event component-based development (TeCBD), we refer to the smallest functional block unit as TeCmp. It is defined in much the same way as a standard component in CBD.

This work is based on component-based software development. In this research, (TeCBD) an emerging software development approach is based on building new software systems from the existing and reusable components. TeCBD involves three stakeholders, TeCmp, TeCnn, and interfaces, which in turn provide, get, or synchronize services. Testing these TeCBD is done first at the component level and then at the assembled unit level.

In this paper, we only focus on the key characteristics of such TeCmp. Individual TeCmp are designed and developed from a hybrid of custom and off-the-shelf (potentially reusable) components. They can be used independently or composed with other TeCmp. In real-time and embedded systems, TeCmp often perform dedicated functionalities under computing and timing constraints as they become more complex and distributed in various environments. Each TeCmp hides its implementation and complexity behind an interface and provides only its functionality to the outside environment, but their interaction and coordination are realized throughout TeCnn.

TeCmp are developed for real-time systems where the logical correctness depends on both the functionality and temporal correctness in a specific environment where the portability should be held to a minimum. Overall, TeCmp describe a syntactically constructive representation where all

 ISBN: 978-1-61208-750-4

tasks are grounded in a set of autonomous functional block units, capturing a common understanding of the application domain at a higher level and according to its semantics.

TeCnn, defined by the protocols, describe the interconnection between TeCmp. That is, they represent a path of interactions between TeCmp and allow transferring data from one Tcomp′s interface to another without compromising the integrity of the data. TeCmp and TeCnn together depict the functionality of the system at runtime.

The overall behavior of TeCnn is to control in a timely fashion the way TeCmp communicate with each other and provide detailed control over the data- and control-flow. Refer to the example of the IoT irrigation application where the system is composed of several TeCmp and TeCnn in Section VII.

IV. Component Comprehension's Challenges in Real-time Systems

We focus our attention on the role of time and modeling which are the most predominant factors when comprehending a real-time system through its source code. In general, real-time systems may involve different disciplines (*i.e.,* IoT, robotic automation), function typically under different real-time computing constraints, and are distributed in various environments. These underlying constraints include, but not limited to timing, liveness, safety, dependability requirements, and evolution of each discipline. Real-time systems are also composed of components that communicate with each other, and each component performs a set of dedicated functions under real-time computing constraints. In a component-based system, components interact with each other in their environment through well-defined interfaces and coordinate protocols by combining each individual component's functionality. Thus, the component-based paradigm entangles both components' computations and services with components' coordination, which turns collectively these autonomous components into a coherent software working application.

First, we focus our attention on the interaction that describes how TeCmp interact rather than focusing on the individual functionalities and services. Furthermore, in this work and in essence of implementing and automating our results, we are aiming at mapping the theory and properties of timed event transitions systems. In particular, timed event automata to TeCmp, an insight in supporting program comprehension for real-time systems. In addition, abstraction, modularity, and modeling are key factors that enable the development of reusable software. We propose a multitude number of layered abstraction views and models which mimic not only common modeling architectural designs, but also improving maintainability and promoting reusability. In the context of this paper, this high layer of abstraction consists of several constructs such as timed event components, ports, timed event connectors, configurations, and interfaces. Importantly, our focus is still on TeCmp and TeCnn. That is, we explicitly express TeCmp and TeCnn, two distinguished component classes, at the implementation level by formally modeling the functionality of TeCmp units and the interaction protocols of TeCnn as timed event finite automata.

The correctness of real-time and embedded systems depends not only on the logical correctness of the computation, but also on the time at which these computations occurred. Furthermore, the structural decomposition of such systems is embodied in their various components and relationship to each other. Thus, there is a need to promote a software space of design alternatives by putting these pieces together, namely a collection of application-specific interfaces, ports, timed event components, port-connectors, and a set of defined real-time constraints. More explicitly, interfaces describe services that TeCmp provide and services they require from other TeCmp, including their compliance with executions. Ports are the access points in TeCmp through interfaces and services. TeCmp can be atomic or composed of layered interactions between a collection of TeCmp that interact with each other providing new functionalities. TeCnn play a primary role in mediating interactions among TeCmp by providing architectural interaction using different techniques such as queries. Furthermore, they provide different type of services such as data transfer, communication protocols, and control transfer. Configurations are a set of associations between TeCmp and TeCnn.

We assume TeCnn can have at least one TeCmp coupled at each of its ports performing operation requests (*i.e.,* data and control). We define three types of interaction interfaces, *get-interface*, *put-interface*, and *syn-interface* where get-interfaces are required and put-interfaces are provided interfaces by TeCmp. However, there may be complicated synchronization constraints between two or more interfaces of a single TeCmp, then we complement TeCmp with a third type of interface that we refer to as *syn-interface*. Two TeCmp, $\mathcal{C}_1$ and $\mathcal{C}_2$, may interact synchronously through syn-interface. Figure 1 shows a timed component-based system with three timed event components, C_1, C_2, and C_3 which communicate through their respective ports, interfaces, and TConn.

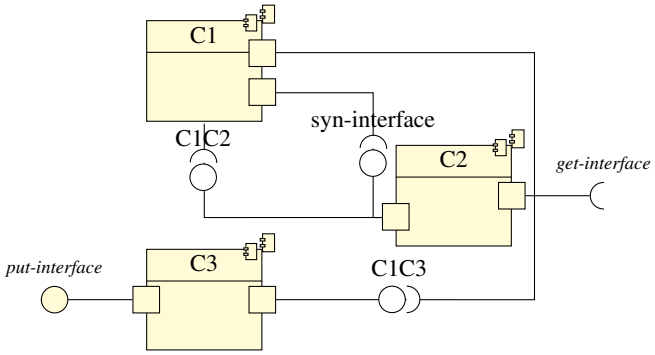

Figure 1. A component-based system with three composed TeCmp, $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ communicating via encapsulated ports/TeCnn, and interfaces.

Now, we can define the following relations between TeCmp. Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be two TeCmp, we define the following TeCmp relationships:

1) TeCmp *Inheritance*

We say two TeCmp, $\mathcal{C}_2$ and $\mathcal{C}_1$, have an inheritance relation, in terms of object-oriented classes, if $\mathcal{C}_2$ inherits all the properties of $\mathcal{C}_1$. In addition, $\mathcal{C}_1$ may have more interaction interfaces and all the inherited interaction interfaces of $\mathcal{C}_2$ work exactly the same way as those of $\mathcal{C}_1$.

2) TeCmp *Association*

We say two TeCmp, $\mathcal{C}_1$ and $\mathcal{C}_2$, have an association relation if they have at least one interaction interface.

3) TeCmp *Aggregation*

We say two TeCmp, $\mathcal{C}_1$ and $\mathcal{C}_2$, have an aggregation relation if $\mathcal{C}_1$ is a subset of $\mathcal{C}_2$. In addition, a single TeCmp can be aggregated by several TeCmp. The aggregated TeCmp has all the interaction interfaces of its TeCmp.

4) TeCmp *Composition*

A composition is the combination of two or more TeCmp at different levels of abstraction to achieve modularity and decomposition of TeCmp using various programming languages or composition tools as defined by the TeCmp infrastructure. Let $\mathcal{C}_1$, $\mathcal{C}_1'$, $\mathcal{C}_2$, $\mathcal{C}_2'$ be four TeCmp. Let the operators $\equiv$ and $\times$ be the equivalence and composition operators in the semantic context, respectively. Then, if $\mathcal{C}_1 \equiv \mathcal{C}_1'$ and $\mathcal{C}_2 \equiv \mathcal{C}_2'$ implies $\mathcal{C}_1 \times \mathcal{C}_2 \equiv \mathcal{C}_1' \times \mathcal{C}_2'$.

5) TeCmp *Encapsulation*

We say that TeCmp $\mathcal{C}_1$ exhibits functional encapsulation if $\mathcal{C}_1$ hides its details while exposing a well-defined interface through its ports. Furthermore, embedded TeCmp may occur at different levels of abstraction and could potentially foresee what we call recursive encapsulation, a fundamental scheme in comprehending programs. We say two TeCmp, $\mathcal{C}_1$ and $\mathcal{C}_2$, have an association relation if they have at least on interaction interface.

The terms association, aggregation, and composition are extended versions of the common terms used in conceptual modeling. In general, the definition of inheritance in this paper is defined in the context of object-oriented programming languages (including C#, C++ and Java). For instance, the inheritance could be considered as covariant, invariant and contravariant in C#.

Abstraction and modularity are key factors in timed component-based framework that enable the development of re-usable software. We start with various and rigorous levels of abstractions and structures that are refined at each stage of the development before mapping them to programming. For instance in timed event components and connectors (TeC&C) model, TeCmp architectural abstractions expose a high-level of the structure of the system, including TeCmp's logical abstractions. On the other hand, TeCnn data- and control-flow abstractions propose categorization spaces of data types and control the flow of imposed conditions. TeCnn communication and synchronization abstraction styles support protocols, and enforce synchronous and asynchronous requests. TeCmp and TeCnn timing abstractions and properties address several issues of real-time systems throughout modeling formalisms.

Component Comprehension's Abstraction

Abstraction can take many forms and dimensions to serve various purposes in software development. In the context of this work, we propose two different levels of abstraction. A horizontal abstraction that studies component comprehension at a very high level of abstraction, such as TeCmp's functionality, ports, interfaces, and TeCnn. However, details and refinements regarding low-level abstractions such as time structures, timed automata, data types, configuration protocols, data structures, and algorithms are performed on a vertical level. In fact, the integration of both horizontal and vertical abstractions reflect an orthogonality at the system and process models, respectively. Partitioning for the purpose of comprehension through various dimensions and abstractions can be found in literature [20] and [21]. Figure 2 views a prism rectangle box with special components, TeCmp, ports, TeCnn, interfaces and a "time event clock".

A prism rectangle box as shown in Figure 2 views special components, TeCmp, ports, TeCnn, interfaces and a "time event clock". Similarly, Figure 3 shows explicitly a series of comprehension views through a high-level horizontal and low-level vertical layers of abstraction. The former layer is composed of constructs such as TeCmp, interfaces, and ports. The latter is composed of constructs such as timed event automata, timed event signature, and source code.

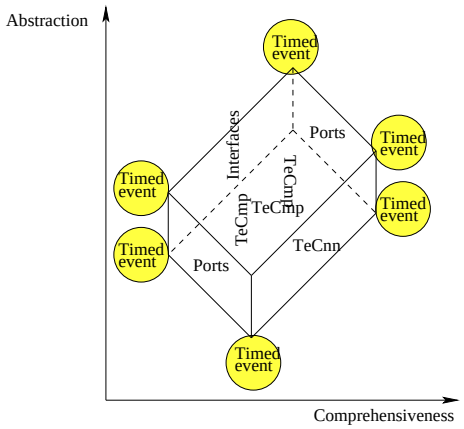

Figure 2. Comprehension dimensions through TeCmp, TeCnn, ports and interfaces.

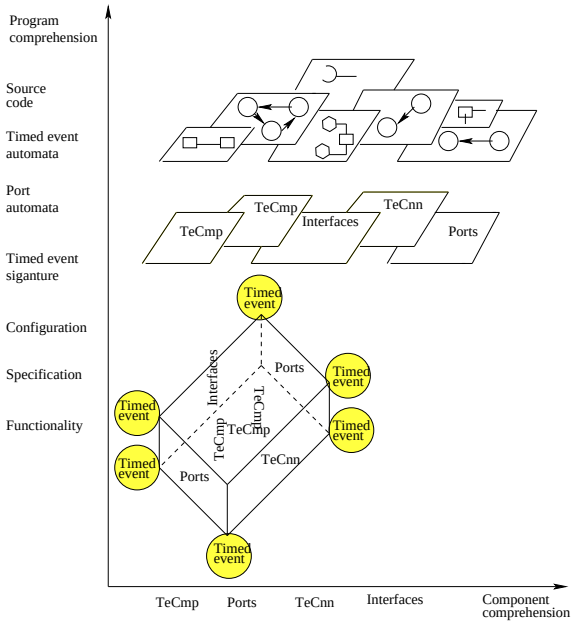

Figure 3. Comprehension dimensions through timed event program.

V. TIMED EVENT COMPONENT AND CONNECTOR MODELS

The coordination and interaction between TeCmp is fully delegated to a special class of component connector that we refer to as TeCnn. An expressive and intuitive way of visualizing TeCnn is to view such a special type of component as a "black box" with some "special code" and a "clock" that allows real-time coordination between the active software timed component entities. Clocks are used to justify timed transitions and sequences of events in such a model. Connectors are modeled as a relation between timed event component streams.

We leverage the time logic and dimension structures of [6] and [22] to describe real-time interactive or concurrent systems in this work. Importantly, we consider the time-dependent behavior of any TeCmp is an important aspect of the system's requirements, enforced by the component itself and coordinated by TeCnn. To develop a uniform timing framework, we consider the absolute time which could be modeled using a global clock.

Let $\mathcal{C}$= $\{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\} \subseteq$ TeCmp be a finite set of timed event component instances where $|\mathcal{C}| = n$. Let $\Omega = (\mathcal{T}, \mathcal{E})$, where $\mathcal{T}$ = $\{t_1, t_2, \ldots, t_k\}$ is a set of points in the time domain and $\mathcal{E}$ = $\{e_1, e_2, \ldots, e_k\}$ is a set of events in the event domain. For convenient, we assume $|\mathcal{T}| = |\mathcal{E}| = k$. Let $\prec$ be a strict partial order precedence relation over $\mathcal{T}$. Let $\mathcal{C}_1(e_1, t_1)$, $\mathcal{C}_2(e_2, t_2)$, and $\mathcal{C}_3(e_3, t_3)$ indicate that $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ are being active on the occurrence of the event e_i at time t_i, respectively where $i = 1 \ldots, n$. We define the timed event dimension structure over TeCmp as a tuple in the form $\mathcal{C}(\mathcal{E}, \mathcal{T})$ that satisfies the following properties:

- (*i*) For all $e \in \mathcal{E}$, if $\mathcal{C}_1(e, t_1) \prec \mathcal{C}_2(e, t_2)$ and $\mathcal{C}_2(e, t_2) \prec \mathcal{C}_3(e, t_3)$ then $\mathcal{C}_1(e, t_1) \prec \mathcal{C}_3(e, t_3)$.
- (*ii*) For all $e \in \mathcal{E}$ and $t \in \mathcal{T}$, $\mathcal{C}_i(e_i, t_i) \not\prec \mathcal{C}_i(e_1, t_i)$, $i = 1, \ldots, n$.
- (*iii*) For all $e \in \mathcal{E}$ and $t \in \mathcal{T}$, if $\mathcal{C}_1(e_1, t_1) \prec \mathcal{C}_2(e_2, t_2)$ then $\mathcal{C}_2(e_2, t_2) \not\prec \mathcal{C}_1(e_1, t_1)$.
- (*vi*) For all $\mathcal{C}_i(e, t)$ and $\mathcal{C}_j(e, t)$, if $\mathcal{C}_i(e, t) \not\prec \mathcal{C}_j(e, t)$ then $\mathcal{C}_i$ and $\mathcal{C}_j$ are interpreted as being concurrent, for all $e \in \mathcal{E}$ and $t \in$, and where $i, j = 1, \ldots, n$.

The external view of the port model is based on the pipe-and-filter architectural style with consists of a set of *data* and *control port* groups. In addition and for various purpose, we assume there is one extra internal group ports that we refer to as *special ports*. The data port group is explicitly divided into input and output data ports. Similarly, the control port group is explicitly divided into input and output control ports. Both the data and control ports are provided by default for each port. However, other types of variables such as monitoring and controlling ports can be an intrinsic part of the internal port depending on the application domain. In the context of this paper, we define a port signature $\mathcal{S}$ as follows:

Definition 5.1: A timed event port signature is a quintuple $\mathcal{S}$ = ($Event$, $Type$, $Data$, $Control$, $Time$), where $Event = \{In, Out, Spec\}$ and In, Out, $Spec$ are the set of input, output, and special ports respectively; $Type$ is a finite set of type names, $Data$ and $Control$ are sets of data and control values, respectively. $Time$ is a set of point structure of time, modeled by a global clock. Moreover, $(In \cap Out \cap Spec) = \emptyset$, and the set of data and control values is disjoint.

In the rest of the paper and for clarity, the terms timed event port signature and port signature are interchangeable. Now, borrowing from the syntax and semantics of components and connectors views, [23] and [24], we formalize the structure of the timed event component and connector model (TeC&C) model by not focusing on the interfaces defined for the ports, but rather on the relation between the different pieces of the TeC&C model.

Definition 5.2: A timed event component and connector TeC&C model is a sextuple structure $\mathbb{CC}$ = ($\mathcal{C}$, $\widehat{\mathcal{C}}$, $\mathcal{P}$, $\mathcal{S}$, δ_p, δ_t) where

- (*i*) $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\} \subseteq$ TeCmp is a finite set of timed event component instances where $|\mathcal{C}| = n$.
- (*ii*) $\mathcal{P} = \{p_1, p_2, \ldots, p_m\}$ is a finite set port instances where $|\mathcal{P}| = m$
- (*iii*) $\mathcal{S} = \{s_1, s_2, \ldots, s_m\} \subseteq$ port signatures is a finite set of port signature instances where $|\mathcal{S}| = |\mathcal{P}| = m$.
- (*iv*) $\widehat{\mathcal{C}} = \{\widehat{\mathcal{C}_1}, \widehat{\mathcal{C}_2}, \ldots, \widehat{\mathcal{C}_q}\} \subseteq$ TeCnn is a finite set of connector instances which are used to capture pathways of events (data transfer flow and control flow) between $\mathcal{C}_i$, $i = 1 \ldots n$. ($|\widehat{\mathcal{C}}| << |\mathcal{C}|$).
- (*v*) δ_p: $\mathcal{C} \times \mathcal{P} \rightarrow \mathcal{C} \times \mathcal{P}$. That is, $\delta_p(\mathcal{C}_i, p_j) \subseteq \mathcal{P}$, for all $i = 1 \ldots n$ and $j = 1 \ldots m$.
- (*vi*) δ_t: $\mathcal{P} \times \mathcal{S} \rightarrow \mathcal{P} \times \mathcal{S}$. That is, $\delta_t(p_j, s_j) \in (\mathcal{P} \times \mathcal{S})$, for all $j = 1 \ldots m$.

The requirements of real-time systems must be able to accommodate real-time timing constraints and discrete/continuous behaviors, such as safety, resources limitations, predictability, and reliability. Thus, the designer must be equipped with modeling formalisms, formal analysis, techniques, and support tools throughout the development process of TeC&C. A well-established modeling formalism to support real-time systems is timed automata [12] that extend finite state automata where transitions are guarded with conditions based on clock variables. Each TeCmp consists of the component requirement specifications, implementations, and interfaces. Consequently, the development of the TeC&C model typically starts with requirements specification which should be written in some formal notations (*i.e.*, formal methods). Thus, it is important to develop a formal description of TeCmp, TeCnn, and TeC&C models for real-time systems. In the following sections, we focus on a series of methodology and automata-based formalisms that capture this collection of timed event interconnected components and connectors. Refer to the example of the IoT irrigation application where the system is composed of several TeCmp and TeCnn in Section VII.

VI. AUTOMATA-BASED COMPONENT AND PROGRAM COMPREHENSION

In the program source comprehension, developers pursue their familiarization effort with TeCmp at various level of granularity, and try to gain an understanding of the program through preliminary evaluations, its structure, and through static and dynamic analysis (or by a combination of both). A drawback is that dynamic analysis can only provide a partial picture of the system based on the developers explorations of the program's behavior through the execution of the system. Static analysis focuses on the source code and extract important information from the program source. The

correctness of a real-time system depends not only on the correctness of the sequence of the events, but also on their time of occurrence. In the following section, we establish approaches and models to understand visualize, and navigate through the source code. First, developers focus on understanding the software as a whole (*i.e.*, component comprehension) avoiding program comprehension whenever possible. Second, developers focus on mental models and visualization during program comprehension inspection activities using timed event automata formalism for comprehending real-time source code and acquire run-time information.

In general, timed transition systems and in particular event transition systems have been extensively studied in the literature [13] [25] [26]. Moreover, both systems have been combined and used practically in the verification, testing, and development of real-time platforms where reliability, safety, and correctness depend to a large extent on the time features. Both time-driven and event-driven computing models are fundamentally important for real-time and embedded systems, given that such systems are reactive by nature. In a time-driven model, computations and actions are triggered by time, either periodically or in terms of deadlines by which computational activities must be completed. In a time-driven model, the state continuously keeps changing as time changes. Thus, the synchronous nature of time guarantees the deterministic behavior of the model. In contrast, in an event-driven model, computational activities or actions are triggered upon occurrences of asynchronous events. We combine time-driven and event-driven models into one unified hybrid system architecture, and propose a real-time model that we refer to as a *Timed Event Automaton* (TeAut). The state of the TeAut continuously changes as time changes and the occurrences of asynchronously generated events forces instantiated state transitions. In consequence, the correctness of real-time system's TeCmp depends not only on the correctness of the computational tasks in the system, but also on the time at which these computations are performed. Let $\mathcal{E}$ and $\mathcal{T}$ denote the event set and time base, respectively, The time domain $\mathcal{T}$ can be modeled as discrete, continuous, or over an interval $[t_l, t_u] \subseteq \mathcal{T}$, $t_l \leq t_u$. In this work, we consider continuous time systems, that is all the variables (*i.e.*, input, output, states) are defined over all possible values of time. In particular, we consider the time domain, $\mathcal{T}$, as the non-negative reals $\mathbb{R}_{\geq 0}$. A timed event $\omega = (e, t)$ over a finite set of events $\mathcal{E}$ and the time $t \in [0, \infty)$ denotes an event $e \in \mathcal{E}$ occurs at time t.

Let $\mathcal{X}$ be a set of finite clock variables (or clocks for short), the set $\Phi(\mathcal{X})$ of clock constraints ϕ over $\mathcal{X}$ is defined by the following grammar:

$$\phi := x \bowtie c \mid \phi_1 \wedge \phi_2 \mid true \mid false$$

where $c \in \mathcal{X}$, $c \in \mathbb{N}$ such that $c \geq 0$, $\bowtie \ \in \{<, \leq, =, >, \geq\}$, and $\wedge$ stands for the *and* logical operator. The precondition clock constraint $\phi \in \Phi$ specifies when the transition is enabled, and the postcondition set $\mathcal{X}_0 \in 2^{\mathcal{X}}$ gives the set of clocks to be reset to zero while all other clocks remain unchanged. A clock valuation represents the values of all clocks in $\mathcal{X}$ at a given snapshot in time.

Definition 6.1: Let $\mathcal{X}$ be the set of clock variables. A *clock valuation* over $\mathcal{X}$ is a function ν from $\mathcal{X}$ to $\mathbb{R}_{\geq 0}$ that maps every clock $x \in \mathcal{X}$ to a non-negative real number.

For $t \in \mathbb{R}_{\geq 0}$, the valuation $\nu + t$ is defined as $(\nu + t)(x) = \nu(x) + t$. For $\mathcal{X}' \subseteq \mathcal{X}$ the valuation is defined as $(\nu[\mathcal{X}' := 0])(x) = 0$ if $x \in \mathcal{X}'$ and $(\nu[\mathcal{X}' := 0])(x) = \nu(x)$ otherwise. We denote by $\mathbb{v} = (\nu_1, \ldots, \nu_n)$ a characteristic vector of clock valuations of the timed automata $\mathcal{A}$. In general, our work partially borrows the concept of time stamps of Leslie Lamport [19], but in particular it is grounded on the foundation of timed automata of Alur [12]. Without loss of generality, a state is defined as a pair $(q, \mathbb{v})$, where $q \in Q$ and $\mathbb{v}$ is a clock valuation at state q.
A time sequence t is a non-empty finite (or infinite) sequence of time values denoted by $t = t_1 t_2 \ldots t_n$ such that $t_i \in \mathbb{R}_{\geq 0}$ and all $t_i's$ satisfies the monotonicity and progressiveness conditions. That is, for all $1 \leq i \leq |t|$, $t_i \leq t_{i+1}$, and for each $t \in \mathbb{R}_{\geq 0}$ there exists $t_i,\ i \in \mathbb{N}$ such that $t_i < t_{i+1}$. If t is infinite then t_i is not bounded for all $i \geq 1$.

We define a finite set of timed events $\Omega = (e_1, t_1)(e_2, t_2) \ldots (e_n, t_n)$ over $\mathcal{E}$ and $\mathcal{T}$, formally denoted as

$$\Omega = \{(e, t)^* \mid e \in \mathcal{E} \cup \{\omega_\lambda\}, t \in \mathcal{T}\}$$

Define $\omega_\lambda = (\lambda, 0)$, where $\lambda \notin \mathcal{E}$ is the null time event to indicate no event has occurred.

Now, we abstract and simulate TeCmp and TeCnn in terms of timed event automata and timed event port-automata, respectively. A timed event automaton induces a timed event transition system. A timed event automaton (TeAut) is a structure defined as follows:

Definition 6.2: A timed event automaton (TeAut) is a sextuple $A = (\Omega, \mathcal{X}, Q, q_0, \Gamma, F)$, where (i) Ω is the finite set of timed events over $\mathcal{E} \times \mathcal{T}$, (ii) $\mathcal{X}$ is the set of clock variables; (iii) Q is the set of states; (iv) $q_0 \in Q$ is the initial state; (v) $\Gamma \subseteq \Omega \times Q \times \Phi(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ is a finite set of transitions; (vi) $F \subseteq Q$ is the set of accepting states.

The values of the clock variables increase monotonically with the passage of time. The next state of a timed event automaton depends on both the event symbol and the values of the clock constraints. In addition, each transition may reset some of the clocks. A transition can only be taken if the current clock values satisfy the time constraints and the event symbol.

Let A be a TeAut and $t \in \mathbb{R}_{\geq 0}$. Define a *timed event requirement specification* A as

$$\mathcal{R}(A) = \{\omega \in \Omega \ : \ \Gamma(q_0, \omega) \in F\}$$

Now, we define timed event port-automata over a single and global clock.

Definition 6.3: A timed event port-automaton (TePA) is a sextuple $\mathcal{A} = (\Omega, \mathcal{S}, Q, \mathcal{P}, \mathcal{X}, \delta)$ where (i) Ω is the set of timed events set; (ii) $\mathcal{S}$ is a port signature; (iii) Q is the set states; (iv) $Q_0 \subseteq Q$ is the set of starting states; (v) $\mathcal{P}$ is the set of all ports; the transition function (vi) $\delta \subseteq \Omega \times Q \times \mathcal{S} \times 2^{\mathcal{P}} \times \Phi(\mathcal{X}) \times Q$.

We extend each timed event port-automaton $\mathcal{A}$ with the powerful primitives of Reo [27] and [28] connectors, a paradigm for communication protocols and composition of software components. Each timed event connector TeCnn via its ports imposes a specific coordination on the active TeCmp, which in turn offer a set of services. The $sync(a, b, e, t)$ time event port-automaton models the Reo primitive that allows

synchronous activities on two ports a and b. Moreover, the synchronous nature of time guarantees the deterministic behavior of the port-automaton. In contrast, computational activities or actions are triggered upon occurrences of asynchronous events. The $lossy(a, b, e, t)$ is similar to the $sync$ primitive, in addition it can have activities through its end, a. The *xor* (a, b, c, e, t) primitive synchronizes a with either b or c. The *fifo* (a, b, e, t) models a buffer with a source port-automaton a and a sink port-automaton b, which are synchronously timed coordinated and asynchronously event triggered. Other operations can be performed on timed event port-automata, such as the product and composition. The desired coordination between two exclusive ports is given as timed event port-automaton. However, the coordination among all TeCmp, as shown in Figure 4, is modeled through individual ports.

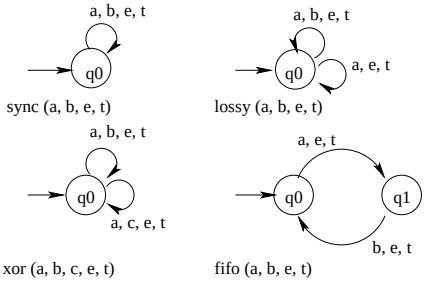

Figure 4. Examples of timed event port automata using Reo's primitives.

VII. IoT Case Study

We describe a case study that has been conducted and implemented on an IoT irrigation embedded system. Overall, the system regulates a water solenoid valve for controlling a drip irrigation system using Arduino and Raspberry Pi infrastructure. For the experiment, we selected and expand the recent IoT project of three graduate students as the basis of our case study by running a variety of experiments to test the proposed theoretical work. The experiment was tested on several events, such as moisture, temperature, and humidity. The system is able to deliver water to the plants based on the moisture of the soil, temperature, and humidity of the day which are obtained through DHT sensors. Importantly, we use a real-time clock that allows the system to set the start of the irrigation system based on the moisture and temperature levels. Furthermore, the system can also start and stop at the specified time intervals to control the water management. In the experiment, the IoT system is controlled by the real-time status of the soil moisture, atmospheric conditions, and on the real time clock to adjust the irrigation scheduling through time intervals. In this IoT-based system, a strong emphasis is put on timed event components of the system and empirical evaluations. The comprehensiveness at both component and program must be sufficiently understood by its developers on performing a broad spectrum of maintenance tasks.

In analogical mapping, our abstract model of study, timed event automata-based components, chas been mapped into the real-time target irrigation domain. That is, soil moisture, temperature, and humidity sensors send real data to the microcontroller, which is considered as the central TeC&C comprehension information gateway. The microcontroller can be monitored and operated via WiFi using a Web browser, or managed by the user through a mobile application. The TeCmp sprinkler controller ensures uniform distribution of water to all part of the plant and it is monitored by the microcontroller. In addition, the TeCmp sprinkler may be switched off and on once the soil moisture sensor has reached the appropriate threshold value. We may consider that DHT moisture, temperature, and humidity sensors are equipped with some ports communicating with various TeCmp. The coordination and interaction between various TeCmp is fully delegated to a special class of component that we referred to as TeCnn. These connectors have no relevant role in the irrigation aspect, but mediate, coordinate, and control interactions among various TeCmp photons. In addition, the data of sensors is displayed in a graphical format, analyzed and visualized by the end-user. This is considered as part of the multi-view learning approaches to perform program comprehension activities. The event domain $\mathcal{E}$ is a set of events in the irrigation domain. That is, $\mathcal{E}$ could be $\{moisture, temperature, wind, humidity, precipitation\}$. When the sensors report that the moisture, temperature, or humidity levels have fallen below the threshold level, the LED light glows, indicating that a timed irrigation event has to be initiate. In addition, the LED lights are also used for other purposes in the context of this irrigation project as summarized below in Figure 5.

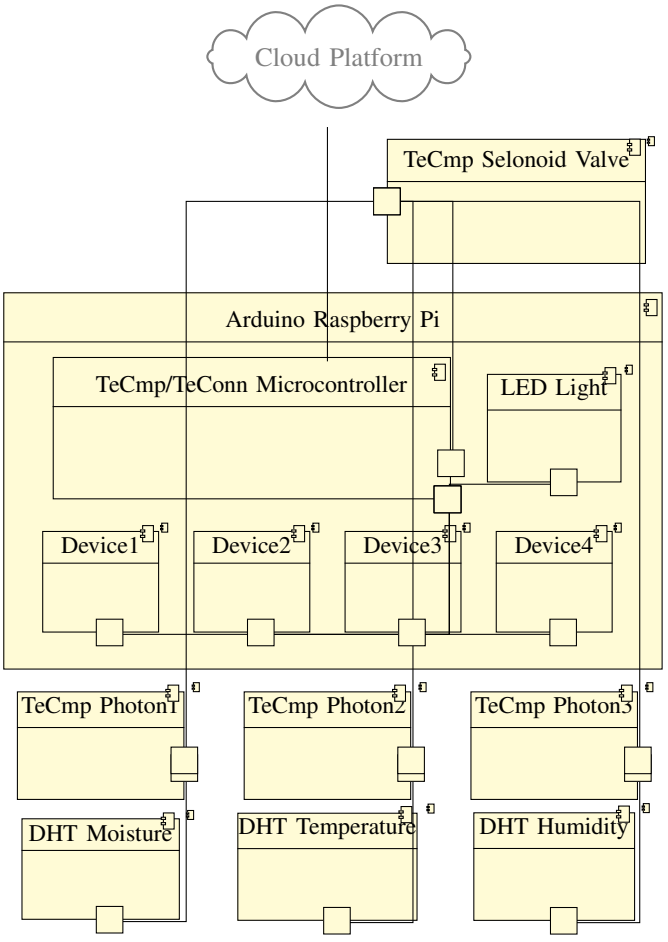

Figure 5. Cloud Control and flow of information in the IoT irrigation System: Soil moisture, temperature, humidity sensors send real-time data to the timed event microcontroller.

 ISBN: 978-1-61208-750-4

VIII. CONCLUSION

Our work revealed an apparent lack of foundations in the literature that relates to program comprehension for real-time and embedded systems. This is an area of research that has not received much attention and could be investigated in various directions. We investigated two timed orthogonal program comprehension paradigms, timed event component and program comprehension, which led to comprehending programs with a greater degree of structure, abstraction techniques, and architecture reconstruction, hence offered a series of potential effectiveness and enhancement in gaining a deeper understanding of program comprehension in real-time systems. First, we mainly rely on architectural levels and time dimensions which have been explicitly targeted in our work and how they are clearly manifested in a real-time systems's implementation. Second, we have examined the relationships between timed event component comprehension and timed event automata-based program comprehension. Such refinement and analysis from component levels to program levels comprehension is significantly represented in the source code. Furthermore, we have performed an empirical IoT irrigation case study in order to complement and provide a qualitative base and characterization of our approach to program comprehension and software evolution. In this work, we validated our theoretical framework on the IoT irrigation application. As a future work, we will investigate this program comprehension paradigm by applying it in various real-time and embedded systems of different domains.

REFERENCES

[1] M. A. Storey, "Theories, Methods and Tools in Program Comprehension: Past, Present and Future," in Proceedings of the 13th International Workshop on Program Comprehension (IWPC '05). Washington DC, DC, USA: IEEE Computer Society, 2005, pp. 181–191.

[2] J. Siegmund, "Program Comprehension: Past, Present, and future," in Proceedings of the 23th Internationa Conference on Software Analysis, Evolution, and Engineering (SANER '16). IEEE SANER, 2016, pp. 13–20.

[3] B. Yuan, V. Murali, and C. Jermaine, "Abridging source code," in Proceedings of the ACM on Programming Languages (OOPSLA 58:1). ACM New York, NY, USA: ACM, 2017, pp. 13–20.

[4] J. Fowkes, P. Chanthirasegaran, R. Ranca, M. Allamanis, M. Lapata, and C. Sutton, "Tassal: Autofolding for Source Code Summarization," in Proceedings of the 38th International Conference on Software Engineering Companion, (ICSE 16). ACM New York, NY, USA: ACM, 2016, pp. 649–652.

[5] D. e. a. Lucia, "Labeling Source Code with Information Retrieval Methods: An Empirical Study," Empirical Software Engineering, vol. 19, No. 5, 2004, pp. 1383–1420.

[6] T. Ben-Nun, A. S. Jakobovit, and T. Hoefler, "Neural Code Comprehension: A Learnable Representation of Code Semantics," in Proceddings of the 32nd Conference on Neural Information Processing Systems (NeurIPS '18, Montreal, Canada, 2018, pp. 3589–3601.

[7] S. Xu, "A Cognitive Model for Program Comprehension," in Proceedings of the 3rd International Conference on Software Engineering Research, Management and Applications (ACIS '05), Montréal, Canada, 2005, pp. 392–398.

[8] T. Reenskaug and J. O. Coplien, "DCI as a new Foundation for Computer Programming," Software Engineering in Intelligent Systems, Springer, vol. 3, no. 5, 2009, pp. 1383–1420.

[9] J. Riling and S. P. S.P. Mudur, "3D Visualization Techniques to Support Slicing-based Program Comprehension," Computer & Graphics, vol. 29, No. 3, 2005, pp. 311–329.

[10] S. Letovsky, "Cognitive Process in Program Comprehension," Journal of Systems and Software, Elsevier, vol. 7, no. 4, 1998, pp. 325–339.

[11] T. LaToza, G. Venolia, and R. DeLine, "Maintaining Mental Models: a Study of Developer Work Habits," in Proceedings of the 28th international conference on Software engineering. ACM, 2006, pp. 492–501.

[12] R. Alur and A. A. Dill, "Theory of Timed Automata," Theoretical Computer Science, vol. 126, no. 2, 1994, pp. 183–235.

[13] A. Fellah, "Timed Event Systems and Automata," in Proceedings of the 13th IASTED International Conference on Control and Applications. Acta Press, 2011, pp. 730–739.

[14] A. Ahmad, P. Jamshidi, and K. Fawad Pahl Claus, "A pattern Language for the Evolution of Component-based Software Architectures," Electronic Communications of the EASST, vol. 59, 2014, pp. 1–31.

[15] O. Le Goaer, D. Tamzalit, M. Oussalah, and A.-D. Seriai, "Evolution Shelf: Reusing Evolution Expertise within Component-based Software Architectures," in Proceedings of the 32nd Annual IEEE International Computer Software and Applications. IEEE, 2008, pp. 311–318.

[16] H. Yin and H. Hansson, "Fighting CPS Complexity by Component-based Software Development of Multi-mode Systems," in Proceedings of the 32nd Annual IEEE International Computer Software and Applications, vol. 2, no. 4. Designs, 2018, pp. 1677–1718.

[17] I. Crnkovic, S. Sentilles, A. Vulgarakis, and M. Chaudron, "A classification Framework for Software Component Models," IEEE Transactions on Software Engienering, vol. 37, 2011, pp. 593–615.

[18] J. Criado, D. Rodriguez-Gracia, L. Iribarne, and N. Padilla, "Toward the Adaptation of Component-based Architectures by Model Transformation: Behind Smart User Interfaces," Software Practice Exp., vol. 45, 2015, pp. 1677–1718.

[19] M. Plakal, D. J. Sorin, A. E. Condon, and M. D. Hill, "Lamport clocks: Verifying a Directory Cache-coherence Protocol," in Proceedings of the 10th Annual ACM symposium on Parallel Algorithms and Architectures. ACM, 1998, pp. 67–76.

[20] R. T. Mittermeir, A. Bollin, H. Pozewauning, and D. Rauner-Reithmayer, "Fighting CPS Complexity by Component-based Software Development of Multi-mode Systems," ACM SIGSOFT Software Engineering Notes, vol. 26, no. 3, 2001, pp. 95–102.

[21] M. Tomgren, D.-J. Chen, and I. Crnkovic, "Component-based vs Model-based Development: a Comparison in the Context of Vehicular Embedded Systems," in Proceedings of the 31st EUROMICRO Conference on Software Engineering, 2001, pp. 95–102.

[22] S. Yu, "The Time Dimension of Computation Models," Where Mathematics, Computer Science, Linguistics and Biology Meet, Springer, Dordrecht, 2001, pp. 161–172.

[23] S. Maoz, N. Pomerantz, and B. Rumpe, "Synthesis of Component and Connector Models from Crosscutting Views," ACM ESEC/SIGSOFT FSE, 2013, pp. 444–454.

[24] S. Maoz, N. Pomerantz, J. O. Ringert, and R. Shalom, "Why is my Component and Connector Views Specification Unsatisfiable?" in Proceedings ACM/IEEE 20th International Conference on Model Driven Engineering Languages and Systems. ACM/IEEE, 2017, pp. 134–144.

[25] T. Brengos, "Behvioural Equivalences for Timed Systems," Logical Methods in Computer Science, vol. 15, no, 1, 2019, pp. 17:1–17:41.

[26] T. Henzinger, Z. Manna, and A. Pnueli, "Timed Transition Systems," in Proceedings of the Real-Time: Theory in Practice, 1991, pp. 226–251.

[27] A. Arbab Reo, "A Channel-Based Coordination Model for Component Fomposition," Mathematical Structures in Computer Science, vol. 14, 2004, pp. 329–366.

[28] F. Arbab, C. Baier, F. de Boer, and J. Rutte, "Models and temporal logical specifications for timed component connectors," Software and Systems Modeling, vol. 6, no. 3, 2007, pp. 59–82.

 ISBN: 978-1-61208-750-4

Limitations of Using Digital BIM Models to Carry out Thermal Analysis

Anabelle Rahhal, Coralie Matthys, Samia Ben Rajeb, Pierre Leclercq
LUCID - Lab for User Cognition & Innovative Design
Quartier Polytech 1, Allée de la Découverte 9, Bât. B52
4000 Liège - Belgium
emails: a.rahhal@uliege.be, cmt@assar.com,
samia.ben.rajeb@ulb.ac.be, pierre.leclercq@uliege.be

***Abstract*— To meet ever more demanding thermal regulations, Building Information Modeling (BIM) and its geometrically semantically enriched building models are presented as a powerful means. In this article, we focus more specifically on the thermal performance of the envelope of existing buildings. Based on the modelling of an existing case, we discuss the potential of extraction and use of the information contained in the digital models to carry out two types of studies: a regulatory certification in the Energy Performance of Buildings (EPB) software and a simulation of energy needs in the Green Building Studio (GBS) software. Through them, we present a panel of the possibilities and the limits of using digital models for this type of study by considering, on the one hand, the quality of the models and on the other hand, the hypotheses governing the methods of the analysis tools.**

Keywords-digital Building Information Modeling (BIM); energy analysis; building data modeling and understanding; reverse engineering.

I. INTRODUCTION

This article is part of a context where digital technology takes a prominent place in the architect's profession. Project design is evolving towards a computerized design thanks to technological advances including BIM tools.

The sector is today subject to ever more numerous requirements in terms of costs, performance and construction techniques. The sophistication of equipment (security, air treatment, home automation, etc.) and the addition of legislative and regulatory constraints increasingly complicate projects and involve dealing with an increasing amount of data [1][2].

Furthermore, another major transformation is underway: the energy transition that results from an international awareness of environmental problems [3]. The building sector has a major role to play in the latter since it is responsible at a European level for 50 % of primary energy consumption and 30 % of greenhouse gas emissions [4]. However, the focus on the energy performance of new buildings is not sufficient and the objectives will not be achieved without an energy improvement of the existing building stock. Thus, energy renovation is one of the main levers for energy saving [5]. In this context, it is essential to have tools that enable to evaluate the environmental impact of a building and to analyze the improvement measures of its energy efficiency in an accelerated manner. It is on the basis of these considerations that states are increasingly encouraging the use of BIM to support the energy transition of buildings.

Regarding the scientific literature, this acronym can refer to several distinct notions. In this article, we focus on BIM as « Building Information Model », i.e., *« ... a digital representation of physical and functional characteristics of a facility, which serves as a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle. »* [6].

The purpose of this article is to identify the possibilities and limits of using digital models to evaluate the thermal performance of an existing building shell. In that respect, we will first study the exploitation potential of the data contained in the BIM models for a regulatory encoding in the EPB software, and we will then analyze these models to perform dynamic thermal simulations in GBS software. These tools are chosen among those available on the market and we have a certain maturity in their use.

This article is structured in four parts. Section II presents a state of the art of the current use of digital models in the building sector. Section III details the methodology established to answer the exploitation issue of these models. Section IV is dedicated to the presentation of results of the implementations that will serve as a support for the discussions developed in Section V.

II. STATE OF THE ART

A. BIM applications

A global survey conducted by the American company McGraw Hill Construction in 2014 revealed that among the 25 applications of BIM, the 3D coordination between construction disciplines and the visualization of design models are the most common cases of uses in the pre-construction phase [7].

The use of BIM is relatively limited for analysis and simulations.

In the energy sector, which is the case of this article, the frequency of use of BIM and digital model barely reaches 25% [8]. However, this study highlights a paradox between the frequency of implementation and the perceived benefit of certain BIM uses, like shown in Figure 1. Indeed, the majority of these are perceived as positive by the respondents, whereas they are not frequently implemented. The main reasons given for their low application rate are the interoperability problems that result from using different file

 ISBN: 978-1-61208-750-4

formats, as well as the resistance of the construction industry towards innovation [9].

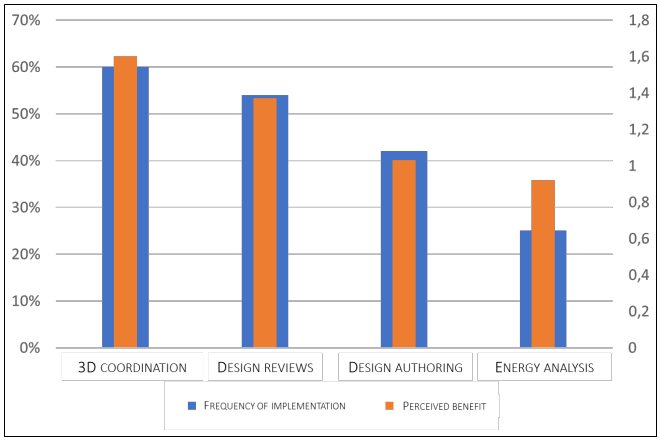

Figure 1. Relation between the frequency and the perceived benefit for implementing each BIM use, adapted from [8].

B. *Interviews*

In parallel with reviewing scientific literature concerning BIM practice in the construction industry, we realized semi-structured interviews at ASSAR Workshop Architects, a BIM precursor architecture office in Belgium. The purposes of these interviews are, on the one hand, to understand how this office implements BIM for renovation projects and, on the other hand, to have an overview of the workflows characterizing projects requiring thermal or energy studies.

These interviews revealed that the digital model is very little exploited for thermal and energy studies while ASSAR is between the most advanced Belgian offices in the implementation of BIM. The interview's answers also indicate that workflows are still very fragmented between architects and energy consultants. In order to evaluate the building's performance, a considerable time is spent on recapturing the project data or adapting the digital model transmitted by the architects.

III. METHODOLOGY

The issues, arising from the review of scientific literature and interviews, guide the work towards studying and modelling a concrete case, in order to evaluate potentialities and limits of using a numerical model to realize thermal analyzes. Figure 2 illustrates the research methodology adopted in this work. First, starting from a modelling protocol and using Revit and Sketchup software tools to produce different models of the case study: "Monolayer Model", "EPB Model" and "Multilayer Model".

Then, quantitative tables were exported from each of these models. The EPB software needed manual encoding to generate its report. Sorting the data collected from the quantitative tables was also necessary to be able to compare the data in Excel software.

Exports in Green Building XML (gbXML) were also accomplished and compared to verify the integrity of the data transfer.

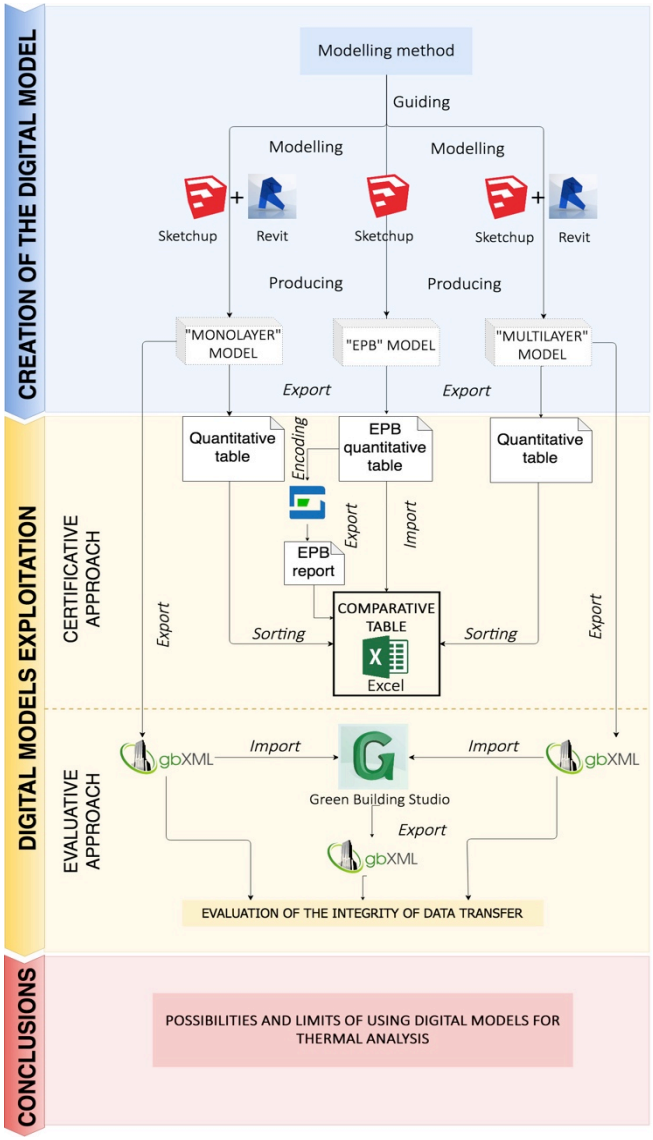

Figure 2. Overall methodological scheme.

A. *Case study*

Our case study is an existing residential building with two parts built respectively in 1900 and 2007 and characterized by:

- a poor thermal performance ;
- a degree of complexity related to the form and constructive hypotheses that involve establishing certain assumptions and simplifications to carry out the modelling;
- a subdivision into five apartments corresponding to six separate thermal zones, identified in Figure 3.

 ISBN: 978-1-61208-750-4

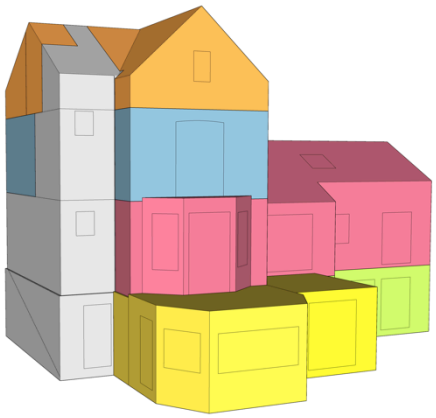

Figure 3. 3D representation of the energetical sectors composing the building.

B. Method

Modeling : The first step of our process is dedicated to building modelling. Autodesk Revit is the chosen modelling tool because it is an object-oriented software, widely used in BIM processes. In addition, it integrates energy analysis and simulation tools useful for our study. Two building models are created based on the available documentation of the existing building by using two different modelling techniques. Figure 4a shows the first technique is the "monolayer" model, which consists of isolating the load-bearing, inner and outer layers of the walls, as distinct elements. Figure 4b illustrates the second technique is the "multilayer" modelling technique, using composite walls, containing several layers of materials.

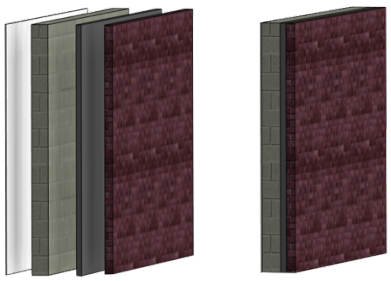

Figure 4. (a) Monolayer modelling. (b) Multilayer modelling.

1) Exploitation of digital models : The created models are then analyzed and exploited according to two distinct approaches.

a) First approach : certificative approach.

The first approach seeks to evaluate the potential of a digital model in a regulatory thermal study in the EPB software. To evaluate this potential, we extract useful data from Revit models. Figure 5 shows that raw data is then sorted and compared with the values calculated and entered previously and manually in the EPB software.

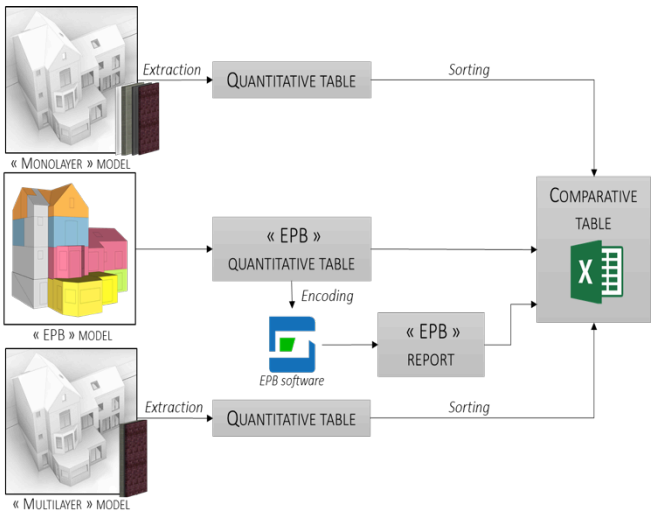

Figure 5. Generation of a comparative table based on the extraction of Revit quantitative tables from “Monolayer” and “Multilayer” models and EPB report.

The studied parameters are :

- heat loss wall surfaces separating the different energy sectors ;
- heat transfer coefficients U and thermal resistances R of the walls, roofs, floors and openings ;
- the heated or conditioned floor surfaces of each energy sector ;
- the volumes of each energy sector.

Most of parameters are extracted from physical models, but analytical energy models are also used to obtain the heat loss surfaces and the volumes of energy sectors (see Table I).

TABLE I. DATA EXTRACTED FROM PHYSICAL AND ANALYTICAL MODELS.

	Heat loss surfaces	**U and R coefficients**	**Heated floor surfaces**	**Volumes of energy sectors**
Physical models	V	V	V	-
Analytical models	V	-	-	V

Figure 6 illustrates the analytical model obtained directly from the physical model and composed of simple surfaces with no thickness. It is interpreted by the and relying on the detailed composition of the construction elements.

The processed data from Revit models are then compared with the data previously encoded in the EPB software, by calculating a variation percentage for each element.

To evaluate how much this variation is detrimental to the EPB study, it is indeed necessary to set thresholds of variation.

 ISBN: 978-1-61208-750-4

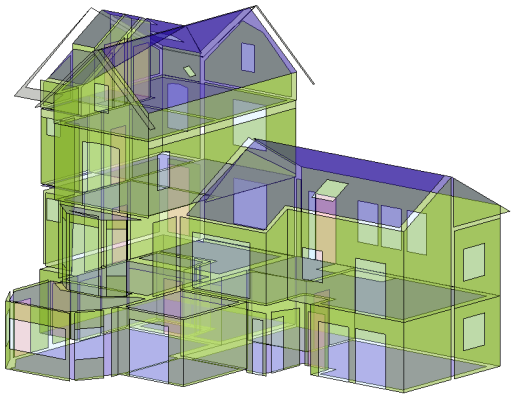

Figure 6. Analytical model.

These are defined in relation to the values of two EPB indicators: Figure 7 shows the specific primary energy consumption and Figure 8 illustrates the net heating energy requirements, which are represented using two scales of values. These values will be used for the analysis of the results.

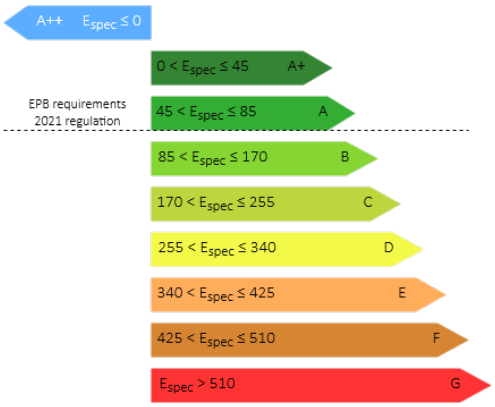

Figure 7. Specific consumption of primary energy.

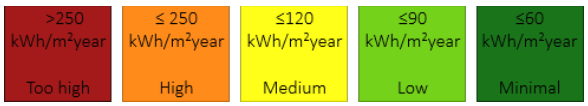

Figure 8. Net heating energy requirements.

To determine the thresholds of variation, we make it possible to fluctuate iteratively and independently the values of the parameters encoded in the initial EPB file. The threshold is then set to the percentage change for which the results indicate that the limit of energy class or performance category of the EPB unit is reached.

b) Second approach : evaluative approach.

The second approach aims to estimate the exploitation potential of the models for an energy simulation. It is performed by GBS software, an Autodesk product, which minimizes the risk of interoperability problems. Figure 9 illustrates the approach that consists of evaluating the integrity of the data transfer upstream and downstream of the simulation, by comparing the exported gbXML Revit files and issued from GBS simulation to the original Revit models.

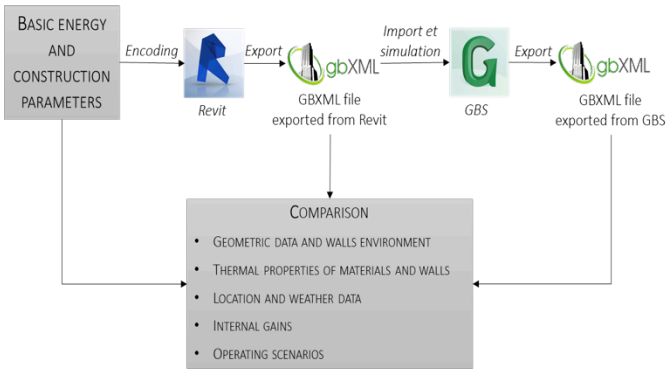

Figure 9. Comparison of the gbXML files and Revit models.

The analyzed data are :

- the surfaces of the walls and their environment (exterior, floor, interior) ;
- the thermal properties of the materials;
- geolocation and meteorological data from the project used to determine heating and cooling design temperatures ;
- internal inputs, which are determined by the energy provided by occupants, equipment and lighting ;
- occupancy and operating scenarios, which include building occupancy times and defining the set point for heating or cooling ; they also include lighting and equipment usage schedules during which heat gains occur.

IV. RESULTS

A. Regulatory study in the EPB software

The results obtained for the first implementation are synthetized in Table II.

TABLE II. SUMMARY OF THE RESULTS FOR THE FIRST APPROACH.

Parameter		Physical models	Analytical models
Heat loss surfaces	Walls	X and -	X
	Floors	V	X
	Roofs	V and -	X
	Openings	V	V
Protected volumes		/	X
Heat floor surfaces		V	/
U and R coefficients		X	/

- = Some values are not defined
/ = Values that cannot be extracted from physical or analytical models
V = Most values are in the variation boundaries
X = Most values are beyond the variation boundaries

 ISBN: 978-1-61208-750-4

The results obtained for the geometric parameters (surfaces and volumes) indicate that the wall loss surfaces extracted from Revit physical models are far from the values calculated for the EPB encoding. Some of them could not even be determined. This difference is explained by the fact that the physical models correspond to the constructive reality of the project. This means that Revit calculates the wall surfaces as they are or will be constructed based on a physical model of the building. However, the regulatory EPB defines the wall surfaces in relation to a conceptual model which simplifies this reality. On the contrary, the floor surfaces coincide because there have not been determined from the floor instances but from surfaces cropped manually in specific plans.

Table II also indicates that most of the geometric data extracted from the analytical models are beyond the thresholds of variation. This difference is explained by the fact that these surfaces are based on an approximative interpretation of the building elements by the calculation algorithm and are not defined the way the EPB method advocates.

In addition, Revit calculates the thermal properties of the walls in a simplified way: it does not take into account the surface exchange heat resistances to determine the U coefficient and it does not distinguish the walls according to their environment whereas these parameters are essential in a regulatory energy calculation.

B. Thermal simulation in Green Building Studio

The results obtained for the second implementation are synthetized in Table III. The results indicate that inconsistencies occurred during data transfer. These have not all been preserved or interpreted during the simulation.

Additional surfaces were superimposed on the openings (doors and windows) and were assigned the same constructive type as the host wall. This results in erroneous geometric data for both gbXML files.

TABLE III. SUMMARY OF THE RESULTS FOR THE SECOND APPROACH.

Parameter	gbXML file exported from Revit	gbXML file exported from GBS
Geometric data	X	X
Walls environment	X	X
Materials and walls thermal properties	V	X
Location	V	V
Weather data	-	X
Internal gains	V	X
Operating scenarios	V	X

- = Non-exported data
V = Successfully exported data
X = Non successfully exported data

Furthermore, the belowground surfaces are defined in Revit based on a horizontal reference plan. The elements under this plane are thus considered buried. However, the land on which the building is located is inclined.

On the contrary, all the thermal properties were exported correctly from Revit to gbXML format. Moreover, the gbXML file analysis of the monolayer model shows that the different layers have been assembled logically. The U values calculated for each layer of material are consistent between the files. However, the thermal transmittance coefficients U of the gbXML file exported from Revit do not take into account the surface exchange heat resistances since this property is not available initially in Revit. After the simulation, the gbXML code analysis of the single layer and multilayer models indicates that all types of doors have been substituted. Green Building Studio has also assigned default thermal properties to additional surfaces that have been created at openings and stairwells when exporting in gbXML.

Location data were successfully retrieved during the gbXML export.

The gbXML file exported from Revit does not contain meteorological data. These ones are normally set automatically by GBS after defining the project location. After the simulation, the analysis of the meteorological data indicates that the heating and cooling design temperatures of the file resulting from the simulation are slightly different from those calculated by Revit. One of the explanations that can justify this difference is the choice of the weather station used to calculate the temperatures for each of the two softwares.

The analysis of the gbXML code exported from Revit indicates that the data on the internal loads are consistent with those encoded in Revit. However, a surface contribution generated by the interior equipment was automatically added during the simulation in Green Building Studio.

The analysis of the gbXML code exported from Revit indicates that operating and occupancy schedules defined in Revit have been exported correctly.

Green Building Studio has, for its part, taken into account four additional scenarios for the simulation: a cooling scenario, a heating scenario, a scenario of domestic hot water use and a ventilation scenario. The equipment operating schedule was replaced by two scenarios: one for the equipment that had previously been defined in Revit, the other one for additional equipment considered by Green Building Studio.

V. LIMITS AND OUTLOOKS OF DIGITAL MODELS FOR THERMAL STUDIES

Our first implementation shows that most quantities extracted from the Revit models cannot be used as they are, but must be sorted in order to compare their values with those of the EPB encoding. The determination of the heat loss surfaces especially requires the use of time-consuming processing methods and involves juggling constantly between the digital models and the extracted tables of quantities to select only the useful elements for EPB calculation.

 ISBN: 978-1-61208-750-4

In the walls quantitative tables for example, it is necessary to keep only those that separate distinct energy sectors. This implies having to remove a large part of the elements contained in the extracted quantities (see Table IV).

TABLE IV. COMPARISON OF THE NUMBERS OF EXCEL LINES FOR THE HEAT LOSS WALLS SURFACES IN THE MONOLAYER AND MULTILAYER MODELS.

Models	Gross statement	Intermediate statement	Final statement
Monolayer	306 lines	84 lines (27%)	42 lines (14%)
Multilayer	284 lines	84 lines (29%)	42 lines (15%)

The data sorting of the single-layer model especially requires the most investment because of the large number of object instances generated. Furthermore, some areas could not be determined on the basis of the constructive model, such as roof areas. Therefore, the wall surfaces of the Revit models obtained on the basis of a material survey are difficult to use for an EPB encoding since they cannot be calculated on the basis of a detailed physical model.

Although analytical energy models have some potential and provide much simpler geometry, their walls areas cannot be encoded in the EPB software. Indeed, the latter are based on an approximative interpretation of the construction elements by the calculation algorithm and are not defined the way the EPB method advocates.

In addition, the first approach highlights the constraints faced by energy consulting firms when working with architects that use digital models. Their work relies on the use of regulatory calculation engines based on historical methods that require manual data inputs. On the one hand, they do not make it possible to directly import the information of a digital model, which prevents the consulting firms from working on the basis of integrated flows. On the other hand, these calculation engines are based on simplifying assumptions whose objective is to facilitate manual encoding. This results in models that are generally far removed from the physical reality of the building.

The second approach highlights a series of inconsistencies upstream and downstream of the simulation in Green Building Studio. Therefore, while it appears to be an interesting tool to perform and analyze various alternatives at the beginning of design, this software is however not very suitable for obtaining an accurate diagnosis of the energy performance of an existing building. There is no doubt that the use of such tools requires analysis, technical know-how and the ability to interpret the results. However, architects and engineering offices need consistent information to guide the design and facilitate the optimization process. Any simulation tool should therefore inform the architect more precisely of the assumptions underlying the results.

VI. CONCLUSION

A. Contributions to research

Our study explores two distinct building modelling methods in order to perform two types of energy studies: a regulatory study and an energy simulation. It allows to define a non exhaustive list of possibilities and limits of using such models, related on the one hand, to the modelling tools (Revit) and to the characteristics of the models themselves (mono and multilayer) and of the other hand to energy analysis tools (EPB and GBS). Finally, exploitation of Green Building Studio allows us to point out the limits of a simulation software and the erroneous conclusions that a user could draw from it.

B. Limits of the research

This work is based on a deep analysis of the digital model and its possibilities and limits for conducting energy studies. The case study was modeled to meet specific energy needs and does not integrate all needed information for all other disciplines in a project. The collaborative aspect of BIM has not been investigated. Nevertheless, this aspect remains one of the intrinsic characteristics of BIM. Exploring BIM as a method of collaboration is not relevant in this work.

A similar finding can be made for the first implementation. Indeed, the exploitation potential of the digital model is not evaluated on the basis of its direct import but on the possibility of using the data that it contains and to be able to encode them manually in the EPB regulatory software, which does not allow currently importing a digital model. However, the objective of BIM is to avoid re-entrying information between the different pieces of software used.

Finally, the method for setting the variation thresholds developed in the first validation process could still be improved. In addition, the setting of these thresholds remains subjective because it is specific to the studied project type and to the initial values of the indicators.

C. Future work

One of the avenues for reflection concerns the computer development of regulatory tools. Indeed, these tools are currently designed to be used at the end of the design as a guarantee of final certification and rely on manual data entry. However, any new or renovation project must comply with the regulations and must therefore be analyzed at its earliest stage, in order to evaluate various possible solutions. It would therefore be interesting to develop interoperability between the BIM digital model and certification softwares such as EPB from exchange formats such as Industry Foundation Classes (IFC) or the gbXML. In addition, the work focused on using Revit software as a modelling tool and Green Building Studio as an energy analysis software. Future work could focus on using other modelling softwares.

 ISBN: 978-1-61208-750-4

ACKNOWLEDGMENT

We would like to give our special thanks to ASSAR l'Atelier Architects, for the time they gave us during the interviews and for enlightening us on the practice of BIM in business.

REFERENCES

[1] N. Bradley and E. Krygiel, Green Building Information Modeling (BIM): Successful Sustainable Design with Building Information Modeling, Hoboken, New Jersey: John Wiley & Sons, 2008.

[2] O. Celnik and E. Lebègue, Building Information Modeling (BIM) & digital model for architecture, building and construction, 2nd ed., Paris: Eyrolles, 2014.

[3] Commission Européenne, *2020 climate & energy package*. [Online] Available from https://ec.europa.eu/clima/policies/strategies/2020_en

[4] S. Trachte and F. Salvesen, "Sustainable renovation of non residential buildings, a response to lowering the environmental impact of the building sector in Europe," in Energy Procedia, vol. 48, pp. 1512–1518, December 2014.

[5] E. Mlecnik et al. Low energy housing retrofit (LEHR), final report. Belgian Science Policy, 2010.

[6] National Building Information Modeling Standard (NBS), Electronic publication: National Building Information Modeling Standard Part-1: Overview, Principles and Methodologies. US National Institute of Building Sciences Facilities Information Council, Building Information Modeling (BIM) Committee, 2007.

[7] McGraw Hill Construction, Smart Market Report : Added Value of Building Information Modeling (BIM) for Construction in Major Global Markets : How architects around the world innovate with building data modelling (Building Information Modeling (BIM)), 2014.

[8] R. Kreider, J. Messner, and C. Dubler, "Determining the frequency and impact of applying Building Information Modeling (BIM) for different purposes on projects," The Sixth International Conference on Innovation in Architecture, Engineering & Construction (AEC), pp. 1-10, University Park, 2010.

[9] S. Beazley, E. Heffernan, and T.J. McCarthy, "Enhancing energy efficiency in residential buildings through the use of Building Information Modeling (BIM): The case for embedding parameters during design," in Energy Procedia, vol. 121, pp. 57-64, 2017.

 ISBN: 978-1-61208-750-4

Industry Case Study: Design Antipatterns in Actual Implementations

Understanding and Correcting Common Integration Design Oversights

Mihaela Iridon
Cândea LLC
Dallas, TX, USA
e-mail: iridon.mihaela@gmail.com

Abstract— **The design of any extensible integration solution involving systems intended to communicate efficiently with one another and/or with data repositories usually begins as a proof of concept or prototype, especially if new technologies and platforms are involved. In some instances, the focus on functional features and tight deadlines lead to inadequate attention placed on non-functional system attributes, such as scalability, extensibility, performance, etc. Many design guidelines, best practices, and principles have been established, and antipatterns were identified and explained at length. Yet, it is not uncommon to encounter actual implementations suffering from deficiencies prescribed by these antipatterns. This paper discusses Leaky Abstractions Mixing Concerns, and Vendor Lock-in, as some of the more frequent offenders in case of system integration. Ensuing problems such as the lack of proper structural and behavioral abstractions are described, along with solutions aiming to avoid costly consequences due to integration instability, constrained system evolution, and poor testability. Moreover, unsuitable technology and tooling choices for database design and release management are shown to lead to a systemic incoherence of the data layer models and artifacts, and implicitly to painful database management and deployment strategies.**

Keywords- integration models; design antipatterns; leaky abstractions; database management.

I. INTRODUCTION

Translating business needs into technical design artifacts and choosing the right technologies and tools, demands a thorough understanding of the business domain as well as solid technical skills. Proper analysis, design, and modeling of functional and non-functional system requirements is only the first step. A deep understanding of design principles and patterns, experience with a variety of technologies, and excellent skills in quick prototyping are vital. Although conceptual or high-level design is in principle technology-agnostic, ultimately specific frameworks, tools, Application Programming Interfaces (APIs), and platforms must be chosen. Together they enable the translation of the design artifacts into a well-functioning, efficient, extensible, and maintainable software system [1].

Designing a solution that targets multi-system integration increases the difficulty and complexity of the design and prototyping tasks considerably, bringing additional concerns into focus. Identifying integration boundaries and how data and behavior should flow between different components and sub-systems, maintaining stable yet extensible integration boundaries, and ensuring system testability, are just a few of such concerns. This paper intends to outline a few design challenges that are not always properly addressed during the early stages of a project. and which can quickly lead to brittle integration implementations and substantial technical debt.

A few recognized design antipatterns and variations thereof are explained here, including concrete examples from actual integration implementations as encountered on various industry projects. Solutions to refactor and resolve these design deficiencies and issues are recommended as well.

Section II will address architectural and integration modeling concerns. The structural aspects discussed in this section range from low granularity models (i.e., data types which support the exchange of data between systems) to large-grained architectural models (i.e., system layers and components). The consequences of designing improper layers and levels of abstractions are outlined, followed by recommendations on how to avoid such pitfalls by refactoring the design accordingly.

In Section III, some antipatterns covered in Section II are extended to the design of the data models and relational databases, discussing also the ability to customize external open-sourced systems that participate in the integration. The focus is later shifted to the management and delivery of the data layer components and artifacts, as databases are an integration concern that goes beyond the data exchanged between the application tier and the data tier. This section intends to explain how the choice of tools and frameworks can have a significant impact on the overall realization, management, and delivery of the integration solution.

Finally, Section IV summarizes the integration design concerns and issues and the recommendations presented in this paper.

II. TIGHT INTEGRATION: LEAKY ABSTRACTIONS AND VENDOR LOCK-IN

A. The Problem Definition

Let's assume the specification of some business needs for building a software system to integrate with - and consume - a third-party service. The exposed data transport models, e.g., Representational State Transfer (REST) models or Simple Object Access Protocol (SOAP) data contracts, are already defined, maintained, and versioned by some external

 ISBN: 978-1-61208-750-4

vendor or entity (the service provider). Note that this scenario can easily be extended further, to integrations with an arbitrary system by means of some third-party APIs that expose specific behavior and data structures as containers for some meaningful payload/data.

Focusing on the data structures rather than behavior, once service model proxies have been generated via some automation, they tend to become part of the design artifacts for the rest of the system. Their use extends beyond the point where they are needed to exchange data with the external application. These models will percolate throughout the various layers and components of the integrating system. It is not unusual to see development efforts proceed around them, with application and business logic rapidly building on top of these data types. Development costs and tight deadlines, and sometimes the lack of design time and/or expertise, are the main reasons leading to this undesirable outcome.

Models exposed by external vendors were not designed with the actual needs of other/integrating systems in mind. External models are characterized by potentially complex shapes (width: number of exposed attributes or properties; depth: composition hierarchy). They cater to most integration needs ("one size fits all"), so they tend to be composed of an exhaustive set of elements to be utilized as needed.

Moreover, allowing these structural characteristics to seep into the application logic layer, beyond the component that constitutes the integration boundary, introduces adverse and unnecessary dependencies to external concerns. Therefore, the system is now exposed to structural instability and will require a constant need to adapt whenever these externally derived models will change. The integration boundary is no longer a crisp and well-defined layer that can isolate and absorb all changes to the external systems – speaking from a data integration perspective.

B. *The Antipatterns*

The lack of proper structural abstractions and allowing integration concerns to infiltrate into the integrating system is a costly design pitfall and is in fact a variation of the "Leaky Abstractions" problem – as originally defined by Joel Spolsky in 2002 [2]. Such deficient abstractions can be identified not only relative to structural models, but also to behavioral models, which could expose the underlying functional details of the software components to integrate with. This will inevitably lead to increased complexity of the current system, jeopardizing its extensibility and its ability to evolve and to be tested independently. Ultimately this results in a tightly coupled integration between the two systems (with strong dependencies on the target of the integration).

Another perspective or consequence of the problem described is an imposing reliance on vendor-specific technologies, their libraries, and even implementations. This problem is also known as the "Vendor Lock-In" antipattern [3]. External system upgrades will necessitate system-wide changes and constant adjustments on the integration side and will impact the overall stability of the system and the integration solution itself.

Examples range from adopting specialized libraries catering to cross-cutting concerns (logging, caching, etc.) to domain-specific technologies (telecom, finance, insurance, etc.). Vendors will encourage integrators to infuse their specialized technology everywhere, leading to entire (sub)systems taking on pervasive dependencies on their technologies, thus making it impossible to separate. Such vendor-dependent architectures must and can be avoided with added effort during the design phase, as described next.

C. *The Solution*

To avoid such scenarios, the design must unambiguously identify the integration boundary and define custom integration models that abstract away any and all structural and behavioral details related to the system targeted for integration. This architectural approach is exemplified in the component diagram in Figure 1. The integration layer should also hide the underlying technology (REST vs. SOAP, message bus vs sockets, etc.) to avoid tight and unnecessary dependencies. An example of defining canonical models based on the "ubiquitous" integration language in case of multi-system integration is presented in "Enterprise Integration Modeling" [4].

Based on the author's experience, designing proper model abstractions proved extremely useful in the case of building custom integrations with real-time systems. For example, Session Initiation Protocol (SIP) soft switches used in telecommunications networks, such as those from Genesys, the leader in customer experience, pertaining to contact center technology (call routing and handling, predictive dialing, multimedia interactions, etc.). In this case, an extensive array of data types, requests, events, etc., are made available to integrators as part of the Genesys Platform SDKs [5]. These facilitate communication with the Genesys application suite – which in turn enables integration with telephony systems, switches, IVR systems, etc. Most of these data types are very complex and heavy, and introduce acute dependencies on the underlying platform, exposing many implementation details as well. Employing code generation

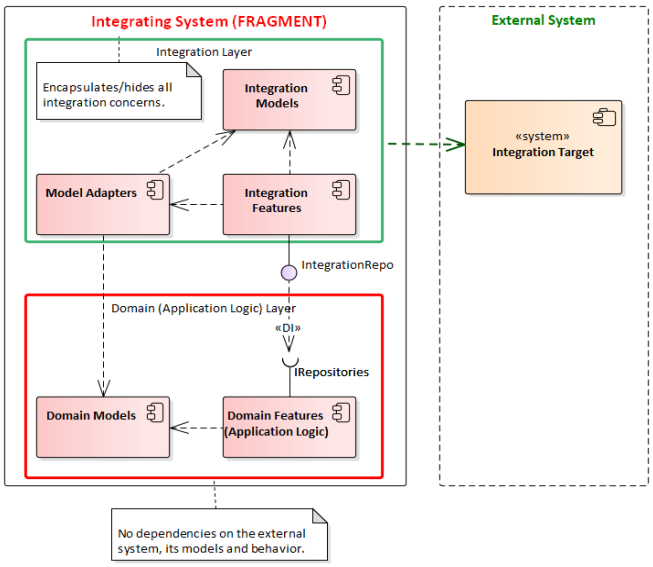

Figure 1. Integration components and the Integration Layer (Adapter) isolating the integrating system from the external system.

and metadata inspection via reflection, for example, simpler connection-less models were designed to mimic and expose only the needed structural details and are currently used in several production systems. Furthermore, defining and realizing the proper architectural isolation layers will ultimately provide independence from vendor-specific platforms for the rest of the system.

For example, considering the integration scenario mentioned above using Genesys' Platform SDK, recently the company (Genesys) has been pushing for a new approach to integrate with their systems, specifically using the Genesys Web Services (GWS) [6], a RESTful API. From an integration viewpoint, this substitution is practically equivalent to switching to a different vendor, as the two integration facilities are based on different technologies (web calls versus direct socket connections) and using completely different models, from both a structural model perspective as well as behavioral and consumption views.

Building an explicit and clean integration layer as shown in Figure 1, when dealing with such a significant change (vendor or technology replacement) changes will be isolated to this adapter layer without any impact on the business domain layer of the integrating system (assuming similar data and functionality). This includes the specifics of the technology used to communicate between the two systems.

Finally, it is noteworthy that four out of the five SOLID design principles [7] substantiate and drive towards the proposed solution:

- Single Responsibility (SRP), from the component and layering perspective,
- Open-Closed, to avoid changing the underlying implementation every time the integration endpoints change,
- Interface Segregation, exposing only the necessary data types for consumption by the business logic layer,
- Dependency Inversion, where the Domain does not directly depend on the external system, its data and behavior, but rather on abstractions – the repository contracts realized by the integration layer.

D. Added Architectural Benefit

Proper design and isolation of the integration components and the use of interfaces and model adapters will enable adequate testing of the custom system without demanding the availability of the external system for integration testing until most defects within the custom system are resolved.

Furthermore, this design approach supports building synthetics that simulate or mock the data and behavior of the external system, providing the means to prototype and test the integration points and functional use cases. Even if only a reduced set of features is synthesized, deferring the needs for actual integration testing can be cost-effective, especially in situations where the external system is a shared resource, perhaps expensive to manage and to access in general.

Employing Dependency Injection (DI) [8], either the real or the mock implementation of the integration contracts can be injected into the Domain layer, making it easy to swap between the two implementations.

III. Data Tier Design, Access, and Management Concerns

One of the most common system integration use cases for many enterprise applications is related to data persistence and access. Integration with (relational) databases that are either part of the custom system or accessible (co-located) components of a third-party system is a pervasive requirement, whether the data tier is needed for storing configuration data, audit/logging, security-related aspects, or to support concrete operational or reporting needs.

This section focuses on several issues related to database design and management, as well as accessing the data itself.

A. 'Inverted' Leaky Abstractions in Data Integrations

1) The Problem

The previous section discussed Leaky Abstractions that result from allowing third-party concerns infiltrate custom systems when designing and implementing an integration solution. The directionality of the "leak", as described earlier, is from the external system into the current one. However, it is also possible to encounter the reverse scenario, when the integration target is open or transparent to the integrators who then take advantage of this fact to develop and apply their own customizations.

Here are two examples:

(a) An Original Equipment Manufacturer (OEM) and/or White Label license of the external system is available to integrators, including access to source code for additional customization and integration options.

(b) The external system contains database(s) accessible to the integrators, on-premise or in a cloud environment, and is open/accessible to change.

In the first example, the same issues and solutions apply, as already discussed in the previous section, only this time from the perspective of the external system. If customization design is not executed properly, software upgrades of the open-sourced third-party system will result in continuous maintenance, or worse, breaking the custom code. Both scenarios will incur high development and system integration testing costs, among other problems.

The rest of this sub-section will focus on the second example, involving third-party databases that are accessible (i.e., open to modification) from an integration and customization perspective.

When expecting and relying on continuous upgrades and patches supplied by the vendor of the external system, it is possible that custom database artifacts (added by the integration provider) will have to be discarded and reapplied, or worse, no longer compatible with the updated system. Moreover, management of database source code targeting the customizations is more difficult if tightly dependent on the elements defined by the external entity/vendor. For example, the custom integration requirements demand two new columns on one of the third-party database tables.

Evidently, with respect to customizations of third-party components (database or otherwise), "Vendor Lock-in" is the status quo as a business-driven need and not a concern here.

 ISBN: 978-1-61208-750-4

2) The Solution

There are several options available and their applicability depends on concrete scenarios and business needs. Ideally, a separate, custom database could be considered, where data collected by the third party system (stored in their databases) would be extracted, transformed as needed, and loaded (ETL) [9]. Detached custom data models are easy to maintain, modify, and version-control by the integration provider. Aligning with the arguments stated in Section II, this approach enforces a well-defined data integration boundary, as shown in Figure 2 below.

Allowing for independent provisioning and evolution of both data models (one provided by the external system and one specifically designed for - and consumed by- the integrating system) will lead to improved extensibility, scalability, performance, testability, and maintainability. With this approach, upgrading the external system will potentially require updating the ETL artifacts and, if needed, some enhancements to the custom database – but both activities can be done in a detached, self-contained fashion.

Further details regarding the management of database artifacts will be discussed later, but one noteworthy benefit here is the freedom from having to maintain (a) partial custom database artifacts (divorced from their context) and/or (b) complete external database artifacts (since the database is a self-contained software system, and should not be divided further into sub-components). The reason why maintaining select/partial database artifacts is undesirable is that from a specification perspective, a database (meaning all its defining artifacts) must be valid, consistent, and complete (as it must also be from a deployment perspective).

If database customizations *must* live in the same database as the one that is part of the external system, a less optimal solution to the Inverted Leaky Abstractions (i.e., the data model), is to expend proper design effort to minimize tight dependencies and attempt to follow - as best as possible - the Open-Closed design principle at the data tier, in the context of system integration and customization.

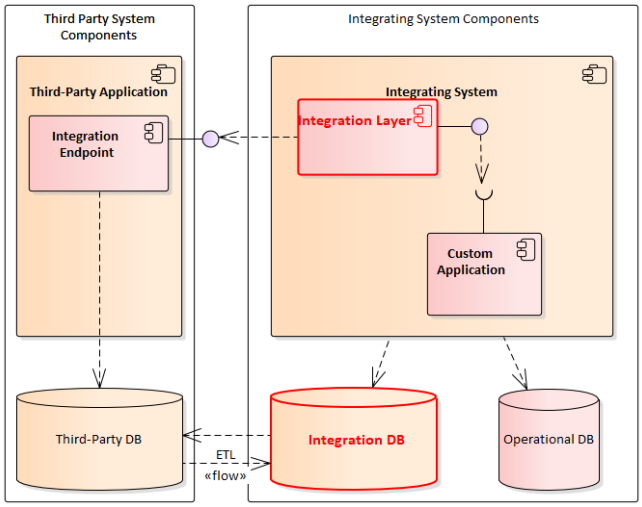

Figure 2. The integration database added to support data integration customizations and to remove direct dependencies on the third-party database.

For example, if the custom integration components require the persistence of new attributes (fields) in addition to the data captured by the external system, rather than modifying the existing third-party tables by adding new columns, association or edge tables should be considered instead, with custom data residing in new, custom tables. Custom views, parameterized or otherwise, should be designed to transform data into a ready-to-consume format (for operational, reporting, or analytical needs).

In this case, the system quality attributes mentioned earlier must however be carefully monitored, especially query performance and scalability. On the downside, database code management will become either (a) fragmented/isolated, by extracting the custom database artifacts from the rest of the database into independent scripts, or (b) more complex, by importing the entire third-party database under source control along with the custom artifacts, in order to preserve its integrity. Subsection D discusses tools that help validate the full database, warning about invalid or broken object references, binding and syntax errors, thus increasing the probability that database deployments will succeed.

B. Mixing Data Modeling Concerns

1) The Problem

Regardless of the targeted Database Management Systems (DBMS) technology, designing the conceptual and logical data models is a prerequisite to the implementation of the physical data models [10]. Beside ensuring that all data elements outlined by the business requirements are accurately represented, non-functional requirements, such as performance, scalability, multi-tenancy support, security (access to data), etc., will also shape the data architecture.

From an application perspective, the database is used to persist the state of the business processes supported by the application, i.e., operational needs, and to support analysis and reporting needs around the stored business data. The concept of Separation of Concerns (SoC) applies here as well but is often ignored. Operational versus reporting concerns are often mixed and data models designed specifically for operational needs are used as such for reporting or analytics purposes, although these models are usually quite different, in terms of how the data is stored and how it is accessed. Yet, it is not uncommon to find a given database used both as the operational as well as the reporting database. As a direct consequence of violating SoC with respect to data modeling (both logically and physically), stability, scalability, extensibility, and performance are the main quality attributes of the system that will be impacted.

An alternate description of this problem is known as the "One Bunch of Everything" antipattern [11], qualifying it as a performance antipattern in database-driven applications, the author aptly pointing out that "treating different types of data and queries differently can significantly improve application performance and scalability."

2) The Solution

Following general data architecture guidelines, the solution is straightforward. In [12], Martin Fowler suggest the separation of operational and reporting databases and

 ISBN: 978-1-61208-750-4

outlines the benefits of having domain logic access the operational database while also massaging (pre-computing) data in preparation for reporting needs. Extract-Transform-Load (ETL) pipelines/workflows can and should be created to move operational data into the reporting database; specifically, into custom-tailored models that cater to requirements around reporting and efficient data reads.

Existing tooling and frameworks can be employed to transform and move data efficiently, on premise or in the cloud (Azure Data Factory, Amazon AWS Glue, Matillion ETL, etc.), for data mining and analytics, for historical as well as real-time reporting needs.

C. Data Access and Leaky Abstractions

1) The Problem

It has been noted [13] that Object Relational Mapping (ORM) technologies, such as Entity Framework (EF) or Hibernate, are in fact a significant cause of data architecture bleed into the application logic, representing yet another example of the Leaky Abstractions antipattern.

Although intended to ease the access to the data tier and the data it hosts, such technologies expose underlying models and behavior to the application tier. In more acute cases – depending on its usage – it also introduces strong dependencies from the domain logic to the data shapes defined in tables, views, and table-valued functions. Entity Framework, for example, while providing the ability to create custom mappings between these data models and the entity models, as designed, these object models are intended to be used as the main domain entities to build the actual domain logic around them. This forces a strong, intertwined yet inadequate dependency between two very different models, targeting different technologies, employed by very different programming paradigms (OO/functional such as C#.NET versus set-based such as SQL). This not only restricts the shape of the domain models, forcing constrained behavioral models to be implemented around them, but also causes data architecture changes to affect the domain and the application logic itself.

Not surprising, Microsoft's EF Core framework in fact discourages against using a repository layer [14] (as prescribed by Evans's DDD [15]) on account that EF itself implements the repository pattern/unit of work enterprise pattern [16] – alas, leading towards a rigid and potentially brittle integration. The reason is that ORM technologies push design and development towards data access logic tangled with the domain logic by encouraging multi-purpose models (domain and data access or data proxies).

2) The Solution

Just as with the integration solution presented in Figure 1, the impact of changes to database models should be constrained to one or two components – those that make up the data access layer, and prohibited from affecting the other application layers, specifically the domain and service layers. Sharing a single model across all layers of the application places unnecessary limitations on the overall design and ultimately on the extensibility and stability of the system.

Although it is uncommon to replace the database technology altogether, sometimes it may be required to replace the *data access* technology due to performance and scalability concerns. Without a proper separation of data access from domain logic and models, such design changes targeting the lower layers of the system architecture are impractical without extensive refactoring of the application.

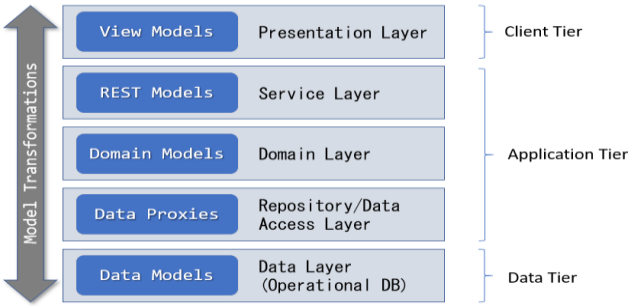

Figure 3. Layered architecture with layer-specific models and model transformations.

In a layered component-based architecture – as shown in Figure 3 above, it is easy and natural to allow each layer to define its own models (darker boxes) and provide adapters to translate from one model to another as data flows through the layers of the application. Although this would seem wasteful at first sight, especially if some models hardly vary from one layer to the next, this approach offers two core benefits. It allows for independent evolution of the models, customizing them to serve very specific needs of the layer they belong to, and keeps the propagation of model changes confined to the corresponding adapter (translation) components.

In case of ORM technologies, the data access layer overlaps with the domain layer, while entity models (shown as data proxies in Figure 3) represent the actual domain models. Interestingly enough, even as ORM is recognized as a Leaky Abstraction, its use is nevertheless encouraged [17], most likely because in unsophisticated implementations, it may be able to deliver acceptable results.

However, as [13] points out, ORM tools can be successfully used "if there is proper separation of concerns, proper data access layer, and competent developers who know what they are doing and really, really understand how relational databases work." Sooner or later, the inherent deficiencies of such technologies, compounded by inadequate implementations due to the lack of understanding of how the underlying technology works, will surface, in most cases under system load and/or when new features are added.

D. Improper Management of Database Artifacts

1) The Problem

Source code, regardless of the language it is written in, is "a precious asset whose value must be protected", as Atlassian's Bitbucket web site states om their "What is version control" online tutorial [18]. All software-producing companies will employ one tool or another for version control. This allows software developers to collaborate, store (or restore/rollback) versions of the software components they build and perform code reviews, and providing a single

 ISBN: 978-1-61208-750-4

stable "source of truth" of the software artifacts they create and release/deploy. As advocated in [19], "source files that make up the software system aren't on shared servers, hidden in folders on a laptop, or embedded in a non-versioned database." Yet, it is rather commonplace to find database implementations that are improperly managed, leading to frustration, bad deployments, making the data tier integration and overall solution delivery unreliable and difficult. There are many online articles and blogs describing such cases.

As encountered by the author, while being engaged as a solution architect and consultant on several projects at various clients, the actual data models and database artifacts were often created and delivered as *ad-hoc implementations* in some arbitrary database, hosted under some arbitrary Microsoft SQL Server instance. Several teams needed these database artifacts: Development for implementation and integration, Quality Assurance for testing, DevOps for deployment. The most common process for deploying this database (fresh install or incremental) to some other environment was to generate and pass around SQL scripts when needed. In somewhat more fortunate situations, these scripts were maintained in some form of source control as SQL/text files, but lacking the ability to validate them or trace the source back to the developer responsible for the actual implementation (in the original database).

So then, where does the "source of truth" for the database definition reside? How can multiple developers work on the database code without overwriting each other's changes and without being aware of the latest updates? How does the organization deliver incremental deployments to any number of target environments? When onboarding new team members, what database code should they be pointed to?

The problems derived from not having a stable, accurate, up-to-date, and complete definition of the database source code, one that is under version control and that can be validated before a deployment, are numerous, acute, and rather obvious. Just as one maintains all other application code under source control, entire solutions composed of many components, why should database implementations not follow the same standards and take advantage of the same acclaimed benefits of code well-managed?

Furthermore, when the database (source) code resides in some database, invalid object references (because someone dropped a column on a table or deleted a stored procedure) will surface only at runtime. Often, changes are made to the database post deployment, even in Production environments, changes that could potentially break the code, or which are at best confined to that environment alone, but without being retrofitted/updated back into the "source code database".

A particularly curious approach to database code management and deployment was encountered on a project that used the Fluent Migrations Framework for .NET [20], self-proclaimed as a "structured way to alter your database schema […] and an alternative to creating lots of sql scripts that have to be run manually by every developer involved." In a nutshell, the tool calls for creating a C#.NET class every time the database schema would change (one class per "migration"). These code files (admittedly, version-controlled) attributed with metadata to identify a specific database update, encapsulate two operations that describe the schema changes: one for a forward deployment ("Up") and one for rollback ("Down"). With a large database, one that evolved considerably over time, with hundreds of artifacts, the number of C# migration files was astounding (thousands). Database changes were published to the target database as part of the application deployment process. Installing the database from scratch would incrementally apply every single "Up" migration specification it finds in these files, following the prescribed update. To maintain sanity, these source code files needed to be named such that the chronological order would be preserved when browsing in the development tool.

However, other more serious problems arise from using this framework, two of them being briefly discussed next.

a) SQL code as C#.NET strings??

Say a new stored procedure must be added; the code is developed and tested from SQL Server Management Studio (SSMS) in some local deployment of the database (assuming the objects the stored procedure is referencing do not change in the interim). Next, a migration file is created, with the "Up" method containing the full (CREATE) stored procedure script, *as a C# string* passed as input argument to the "Execute.Sql" method call.

The major and obvious problem here is the inability to validate SQL syntax and semantics and SQL object references when represented as indiscriminate plain strings, subject to typing errors.

b) No database source code??

Unless deployed on some SQL Server instance, it is impossible to even begin to understand the structure of the database, even the structure of individual objects. The data models and data logic are scattered, fragmented (across many C# files), impossible to validate (syntactically or otherwise) from where the database "source code" is stored.

Moreover, a given database object, say a table for example, can change any number of times, each change being captured in a different source file, with no unified, single view of what that table looks like, what the shape of the data is, with all its columns and corresponding types, with its keys and indexes, constraints and triggers, if any. This problem extends to all database objects, not just tables.

The data models (the source code artifacts) are practically non-existent, difficult to comprehend, and cannot be validated until they are deployed. The result is a total and indefensible representational incoherence afflicting the most important component of a data-dependent enterprise system.

2) The Solution

There are various software tools available to address this problem. Both Microsoft and Redgate, for example, provide excellent tooling for developing relational databases, managing database artifacts under source control, facilitating change management and incremental deployment, generating manual update scripts (when automated deployment is constrained), and more.

Microsoft's SQL Server Data Tools (SSDT) [21] is a development tool, available since 2013, using the Data-Tier Application Framework (DacFx). It facilitates the design and

 ISBN: 978-1-61208-750-4

implementation of SQL Server and Azure SQL databases, as well as database source control and incremental deployment, all integrated under the Microsoft Visual Studio development environment.

A version-controlled database project contains all distinct database objects as individual files, and it *must* compile – targeting a specific SQL Server (or Azure) database version – before it can be deployed anywhere. Developers can check out individual objects (files) to change as needed or can add new objects using the provided templates. Just as one can see the entire schema of a database in SSMS, similarly, anyone can see and browse these objects in Visual Studio.

Tools like SSDT are also capable of identifying the changes (delta) between the source and the destination database in order to create the appropriate deployment scripts, and ultimately allowing rapid and valid delivery of database changes to any environment.

It is questionable to store Java or C# code in SQL scripts, with artifacts/classes shredded and reduced to SQL NVARCHARs, scattered in an arbitrary number of stored procedures (equal to the number of updates effected upon that class), and passed around to call other stored procedures (via EXEC statements). The reverse scenarios should be equally unacceptable. Treating the database as a proper software implementation artifact is imperative.

IV. CONCLUSION

This paper aimed to raise awareness about certain design challenges that, when not addressed early and properly, will lead to deficient architectures and rigid solutions concerning various aspects of system integration, as often encountered in practice.

When the design of software systems follows some basic guidelines and principles (SOLID), the resulting architecture will allow the system to be easily built, modified, and extended. In case of system integrations and customizations, violating these principles and particularly the multi-faceted Separation of Concerns design rule, leads to unmanageable and highly complex systems that do not scale well, cannot be extended or modified easily, with tight dependencies on external components and overall brittle integration solutions.

Many design antipatterns have been catalogued and well documented; yet deficient architectures are encountered quite frequently, leading to high technical debt and unhappy stakeholders. This paper discussed “Leaky Abstractions”, “Mixing Concerns”, and “Vendor Lock-in” antipatterns – from the perspective of concrete industry examples, as encountered and worked on by the author.

Concrete approaches that address these problems to help refactor and realign the design according to best practices and principles were elaborated, explaining how they lead to scalable, extensible, testable, efficient and robust integration solutions.

Relational database design and management concerns were also presented, with focus on data model design, data access practices, and management of database artifacts. The consequences of improper tooling and frameworks were briefly covered, and a solution discussed.

ACKNOWLEDGEMENT

I would like to thank my husband and long-time mentor, Chris Moore, for his indefatigable guidance and for sharing the extensive technical knowledge and experience he possesses and masters so adeptly.

REFERENCES

[1] R. Martin, “Clean Architecture,” Prentice Hall, 2018, ISBN-13: 978-0-13-449416-6.

[2] J. Spolsky, “The Law of Leaky Abstractions,” [Online] Available from https://www.joelonsoftware.com/2002/11/11/the-law-of-leaky-abstractions/ [retrieved: June, 2019].

[3] SourceMaking, Software Architecture AntiPatterns, [Online]. Available from https://sourcemaking.com/antipatterns [retrieved: June 2019].

[4] M. Iridon, “Enterprise Integration Modeling,” International Journal of Advances in Software, vol 9 no 1 & 2, 2016, pp. 116-127.

[5] Genesys, “Platform SDK,” [Online]. Available from https://docs.genesys.com/Documentation/PSDK [retrieved: June, 2019].

[6] Genesys, “Web Services and Applications,” [Online]. Available from https://docs.genesys.com/Documentation/HTCC [retrieved: June, 2019].

[7] G. M. Hall, “Adaptive Code via C#: Agile coding with design patterns and SOLID principles (Developer Reference),” Microsoft Press, 1st Edition, 2014, ISBN-13: 978- 0735683204.

[8] M. Seemann, “Dependency Injection in .NET,” Manning Publications, 1st Edition., 2011, ISBN-13: 978-1935182504.

[9] Microsoft, “Extract, Transform, and Load (ETL), ” [Online]. Available from https://docs.microsoft.com/en-us/azure/architecture/data-guide/relational-data/etl [retrieved: June, 2019].

[10] G. Simsion and G. Witt, “Data Modeling Essentials,” Morgan Kaufmann; 3rd edition, 2004, ISBN-13: 978-0126445510.

[11] A. Reitbauer, “Performance Anti-Patterns in Database-Driven Applications,” [Online] Available from https://www.infoq.com/articles/Anti-Patterns-Alois-Reitbauer/ [retrieved: September, 2019].

[12] M. Fowler, “Reporting Database, ” [Online]. Available from https://martinfowler.com/bliki/ReportingDatabase.html [retrieved: September, 2019].

[13] V. Bilopavlovic, “Can we talk about ORM Crisis?”. [Online] Available from https://www.linkedin.com/pulse/can-we-talk-orm-crisis-vedran-bilopavlovi%C4%87 [retrieved: June, 2019].

[14] Jon P. Smith, “Entity Framework Core in Action,” manning Publications, 2018, ISBN-13: 978-1617294563.

[15] E. Evans, “Domain-Driven Design: Tackling Complexity in the Heart of Software,” 1st Edition, Prentice Hall, 2003, ISBN-13: 978-0321125217.

[16] M. Fowler, “Patterns of Enterprise Application Architecture,” Addison-Wesley Professional, 2002.

[17] M. Fowler, “OrmHate,” [Online]. Available from https://martinfowler.com/bliki/OrmHate.html [retrieved: June, 2019].

[18] Atlassian, “What is version control, ” [Online]. Available from https://www.atlassian.com/git/tutorials/what-is-version-control [retrieved: September, 2019].

[19] P. Duvall, “Version everything,” [Online]. Available from https://www.ibm.com/developerworks/library/a-devops6/ [retrieved: September, 2019].

[20] “Fluent Migrations Framework for .NET,” [Online]. Available from https://fluentmigrator.github.io/ [retrieved: September, 2019].

[21] Microsoft, “SQL Server Data Tools,” [Online]. Available from https://docs.microsoft.com/en-us/sql/ssdt/sql-server-data-tools [retrieved: September, 2019].

 ISBN: 978-1-61208-750-4

Data Science as a Service

Prototyping for an Enterprise Self-Service Platform for Reproducible Research

Steve Guhr
NetApp Deutschland GmbH
Berlin, Germany
e-mail: steve.guhr@netapp.com

Detlef Amendt
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: amendt.detlef@mh-hannover.de

Jan-Hendrik Martenson
NetApp Deutschland GmbH
Hamburg, Germany
e-mail: jan-hendrik.martenson@netapp.com

Benjamin Schantze
NetApp Deutschland GmbH
Hamburg, Germany
e-mail: benjamin.schantze@netapp.com

Hans Laser
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: laser.hans@mh-hannover.de

Svetlana Gerbel
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: gerbel.svetlana@mh-hannover.de

Jannes Gless
Center for Information Management
Hannover Medical School
Hannover, Germany
e-mail: gless.jannes@mh-hannover.de

***Abstract*—A data scientific process (e.g., Obtain, Scrub, Explore, Model, and iNterpret (OSEMN)) usually consists of different steps and can be understood as an umbrella for the combination of different most modern techniques and tools for the extraction of information and knowledge. In this paper, we show a prototypical implementation for the efficient use of available compute center resources as a self-service platform on enterprise technology to support data-driven research. Scientific requirements for reproducibility and comprehensibility are to be taken into account.**

Keywords—Data Science as a Service; reproducible research; enterprise information technology; self-services; cloud infrastructure; data science platform.

I. INTRODUCTION

Technology should be available to everyone. That is one of the reasons why companies build services. One of the key aspects to consider when building service portfolios should be to make it as easy and consumable for the end-user as possible. The main challenge is to "carry" those known tools and solutions into a scalable and powerful platform that can be provided with enterprise technology to warrant Service-Level-Agreements (SLA) from a central enterprise Information Technology (IT).

A. The Data Science Process

To obtain information (e.g., based on patterns) for relevant business decisions from data of heterogeneous data sources, a classical multi-stage process for data preparation and analysis is used, the so-called data mining process [1]. Data science, on the other hand, can be understood as an umbrella for the combination of various state-of-the-art techniques for the extraction of information and knowledge (so-called insights) to develop data-based applications and, thus, automating processes. One approach to describe the individual steps for the data science process is **O**btain, **S**crub, **E**xplore, **M**odel and I**N**terprenting (OSEMN) [2]. In the Obtain step, for example, query languages are required for databases that can be extracted in various formats. Python [3] and R [4] encapsulate the otherwise heterogeneous data query tools (e.g. Structured Query Language (SQL), eXtensible Markup Language (XML), Application Programming Languages (API), Comma Separated Value (CSV) and Hybrid File System (HFS)). Classic database techniques such as Extract Transform Load (ETL) process can be used in the cleanup step (Scrub). Computer languages like Python and R or application suits like SAS Enterprise Miner [5] or OpenRefine [6] can also be used to transform data. To examine the data (Explore) languages like Python or R specialize in particular appropriate libraries (e.g. Pandas [7] or Scipy [8]). In this step, however, familiar players from the business

 ISBN: 978-1-61208-750-4

intelligence world (e.g. Rapid Miner [9] or KNIME [10]) can also be found for data-wrangling. To build a model, there are again specialized Python libraries like "Sci-kit learn" [11] or CARET [12] for R. Other tools like KNIME or Rapid Miner find reuse in this step as well. Finally, for interpreting the model and the data, as well as evaluating the generalization of the algorithm, tools for data visualization are reused (e.g. matplotlib [13], Tableau [14] or MS Power BI [15]). In summary, it means that, for the many single steps in OSEMN, many different tools can be necessary.

B. Reproducible Research

In the domain of computer-aided data-driven science, researchers today use common libraries and tools. Researchers often choose free and open source tools [16].

The reproducibility and repeatability of the research results and the description of the specific runtime environment in which the results were created are described in the respective publications either not at all or only textually [16][17]. An important factor in the publication of the scientific work is the reproducibility of the research results [17][18].

The data principles published in 2016 define fundamentals that research data and research data infrastructures should meet in order to ensure sustainability and reusability [19]. The scientific data should therefore be **f**indable, **a**ccessible, **i**nteroperable and **r**eusable (FAIR data principles). In a highly simplified way, data and services should be stored in central data repositories using suitable metadata (F), taking into account aspects of long-term archiving (A), and should be able to be exchanged and interpreted (semi-)automatically (I) and thus be comparable and reusable (R).

Results of systems research show that open source tools in particular are suitable for reproducibility requirements. Although Docker was introduced primarily for enterprise needs and Web application delivery, it provides solutions for virtualization, platform portability, reuse, sharing, archiving, and versioning [17] for the scientific community.

The use of tools such as Jupyter Notebooks (jupyter.org) enables semantically interoperable publication of program code, including through the use of the IPYNB format [16]. Jupyter Notebooks supports workflows in the fields of scientific computing, astrophysics, geology, genetics and computer science [16]. Various applications and programming languages (e.g. Python, R) offer interfaces to Jupyter Notebooks [17][20]. Jupyter gathers many valuable tools that are needed in the steps of the OSEMN process model.

C. Aims of the project

NetApp is one of the leading independent data management providers [21] and has been helping organizations store, manage, secure, and leverage their most mission-critical data assets for more than 25 years.

The Center for Information Management (ZIMt) of the Hannover Medical School (MHH) centralizes operative systems and is a service provider especially for the areas of research and teaching, clinic and administration. The ZIMt operates a class tier 3 [22] computing center at the MHH and is ISO 9001:2015 certified. The centralization of applications to support the scientific field is a strategic goal of the MHH.

The goal of this project is to establish an easy to maintain and cost-efficient infrastructure to support the data-driven research and the implementation in the existing data center infrastructure. Thereby, the requirements according to FAIR and the operation of enterprise-level applications will be considered.

The rest of the paper is structured as follows. In Section II, we describe the methods used to address the above mentioned challenges. In Section III, we describe the results achieved referring to the main issues. Section IV concludes this work addressing open issues and next steps.

II. METHODS / APPROACH

For the operation of Jupyter Notebooks in the data center, the open-source environment JupyterHub Notebook Server is used. It enables users to access computing environments and resources without bothering users with installation and maintenance tasks.

Docker is a technology that abstracts applications from the underlying operating system to gain portability for software solutions wrapped in so called "Containers". With a standardized environment (the Docker software/binaries running on many different operating systems today) built for containers one can use individual application manifests on different sites (e.g. on-premises as well as in a public cloud infrastructure provided by Google, Amazon Web Services (AWS) or Microsoft Azure) without the need to change any code.

JupyterHub uses Docker as the foundation for deploying Jupyter Notebooks. The Jupyter Docker Stacks project [23] will provide standardized Jupyter Notebooks environments for various applications using Docker images, including pre-configured environments for data science use. For special requirements of the development environment, it is possible to offer further images that can be individually modified to the needs of the user.

Docker containers are more convenient to deploy, easier to manage, minimize overhead in resource usage, and are therefore more efficient than traditional virtual infrastructures [24]. The provisioning of environments (e.g., through containers) in the academic field can be very extensive, thus increasing the burden on the operators and maintainers of the environment. Automating the provisioning and orchestration of the environment is highly recommended.

Running applications in containers also does not automatically solve the challenge of protecting those applications against failures (e.g. hardware outages or resource bottlenecks on this one server we are working on). Even though containers are encapsulated on top of an operating system, there might be issues with the underlying host running the container - therefore, an additional software layer taking care of resource scheduling and availability of our Jupyter Notebooks is needed.

 ISBN: 978-1-61208-750-4

Kubernetes has been used as an open-source solution to orchestrate, automate and meet those high availability requirements for container-based infrastructure [25]-[27]. It is a widely used and proven technology for delivering services like Jupyter. Because Kubernetes is designed to host an enormous number of applications with minimal overhead, it is perfect for many Jupyter Notebooks and other potential applications within the science ecosystem [24][28].

For more flexibility in design, a hypervisor (VMware) was used to deploy the hosts for container orchestration based on Docker and Kubernetes. Terraform and Ansible are fully automated - Terraform creates the virtual machines within the hypervisor, Ansible handles the installation of the packages and configuration of the hosts (configuration management). To avoid inconsistencies in the configuration, the DevOps (software Development and information technology Operations) paradigm applies to be an "immutable" infrastructure, where every change in the ecosystem leads to a completely new deployment of the entire stack [29].

Docker containers are ephemeral, which means that they do not persist data after termination of their life cycle. Data storage must be ensured and is therefore required to secure the data beyond the life cycle of the container.

In terms of the reproducibility of the results, the collected data becomes the most important asset in the process chain - since a robust and highly available architecture is to serve as the basis, a NetApp storage system is used to store the data. The central storage system consolidates data in the data center. This prevents, on the one hand, the storage of data on terminals and on the other hand, enables a more effective backup.

The Network File System (NFS) was chosen as a protocol for the connection to the storage. This decision is based on being able to make the stored data available outside of the data science platform. NFS can be used by several clients at the same time with write access. This integration into conventional infrastructures simultaneously allows the usual access via Explorer or Shell.

Within Kubernetes, so-called storage classes can be used to provide storage from an external source - in this case, a container gets persistent storage space to store data beyond its own life cycle, thereby making the data reusable. Trident [30] is an open-source storage orchestrator for Kubernetes provided by NetApp and can be used to add persistence to the Jupyter Notebooks. Since each notebook is to be provisioned individually for a particular user, each user also receives an exclusive area for data storage. This is made possible by the use of authentication within the ecosystem, so each user must provide credentials to be accepted.

Even though we are in a “proof of concept” phase of the project, we wanted to integrate authentication methods from the very beginning to control access to the platform while obtaining compliance in terms of security. At the MHH, a local security area for managing objects (e.g. usernames, computers, printers, etc.) is implemented as a domain via Microsoft Windows Active Directory. For the centralization of user IDs administration using Role-Based Access Control (RBAC), the authentication to the Active Directory using Lightweight Directory Access Protocol (LDAP) implementation of JupyterHub was accessed.

III. RESULTS

This prototypical implemented infrastructure allows the end-user (students, scientists) to easily use Jupyter Notebooks. Using the Docker-based approach, the description of the runtime environment required for the research approach can be fixed using the Docker-specific tagging option and stored in a manifest in a comprehensible and interoperable manner for publication [17].

If the researcher chooses a work environment based on Jupyter Notebooks, necessary work steps and results can be saved together with the notebook [16][20]. The basic condition to support requirements as described in FAIR could be taken care of.

By default, configurations of Jupyter Notebooks are offered and further evaluated via the JupyterHub Spawner (Basic and Data Science). Additional libraries or tools, which may not be included in the standardized environment, can be installed flexibly to the current runtime environment in their own separate area. Jupyter Notebooks, therefore, offer the possibility to use multiple tools without having to change the environment. Requirements for various tools, such as those required in processes such as OSEMN, can be supported by Jupyter Notebooks.

The isolation of the specific workspace of a researcher can be solved by using Docker. Regardless, central compute resources can be shared and used efficiently across multiple environments.

The operator of the infrastructure (ZIMt) achieves a work facilitation through the selected reference architecture (see Figure 1) by automating the provision of resources for the users (researchers). Because JupyterHub is deployed through Docker, the branding requirements for maintaining the Corporate Identity can be easily met. By using the existing Active Directory, user access can be controlled centrally. Authentication via LDAP simplifies logging on to the system, since no separate access data needs to be maintained.

By provisioning the required storage area at runtime, resources can be provided centrally and efficiently. The persistence of research data outside the Docker container runtime on the existing enterprise storage system could be efficiently solved by using Trident (see Figure 1).

Each time a user logs on to the home page, the system checks to see if the user already has a Jupyter notebook created in the past. In this case, he or she will be redirected to this existing environment. Otherwise, a new notebook is created (in the form of a new container) and new storage space is provided (because that user did not exist before).

 ISBN: 978-1-61208-750-4

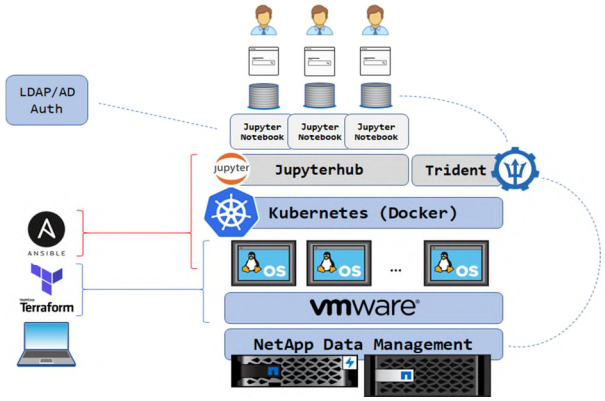

Figure 1. Prototyped Architecture to provision Jupyter Notebooks on Enterprise Technology

IV. CONCLUSION

In this paper, we showed a prototypical implementation for the efficient use of available compute center resources as a self-service platform on enterprise technology to support data-driven research.

The use of the predefined configurations of the Jupyter Notebooks is initially limited by the images. It will show with prolonged use of the service if the provision of additional images would be useful.

For the prototype implementation of this infrastructure, one Kubernetes master with two Kubernetes workers was deployed. For productive operation, at least two Kubernetes masters should be used to meet the requirements for failure safety. In the event of a disaster recovery scenario and the loss of the complete Kubernetes cluster, the storage volumes that are deployed through Trident must be manually reconnected. An automatism for restore procedures would still have to be created. In an emergency, the administrator can migrate the contents of the corresponding volume through NFS.

Existing and established methodologies like OAuth [31] or OpenID [32] would have required additional components to be available. However, those security concepts will be considered in later phases of the project after the initial thesis is validated (if this software stack is suitable for the use case at all).

If the JupyterHub internal database is lost, the link to the pod and, thus, to the individual runtime environment is lost and needs to be restored. To use Jupyter notebooks on existing HPC infrastructures, the use of the Jupyter Enterprise Gateway [33] needs to be evaluated.

As already mentioned before, this project implemented only a proof of concept to demonstrate the feasibility of IT operations by combining common data science tools with enterprise architecture. Beyond Jupyter Notebooks related to machine learning, tools such as Airflow [34] or Pachyderm [35] (as a platform solution) could be used for pipelining and automation in the next development stages. These tools could support process models such as OSEMN, as well as aspects of reproducibility and re-usability.

REFERENCES

[1] C. Li, “Preprocessing Methods and Pipelines of Data Mining: An Overview”, CoRR, 2019, arXiv:1906.08510

[2] H. Mason and C. Wiggins, “A Taxonomy of Data Science”, dataists.com, 2010, http://www.dataists.com/2010/09/a-taxonomy-of-data-science/, last accessed 2019/07/22

[3] Python Programming Language, 2019, https://www.python.org, last accessed 2019/10/23

[4] The R Project for Statistical Computing, https://www.r-project.org, last accessed 2019/10/23

[5] SAS® Enterprise Miner™, 2019, https://www.sas.com/en_us/software/enterprise-miner.html, last accessed 2019/10/23

[6] Open Refine, http://openrefine.org/, last accessed 2019/10/23

[7] Pandas Python Data Analysis Library, https://pandas.pydata.org/, last accessed 2019/10/23

[8] Scipy, 2019, https://www.scipy.org/, last accessed 2019/10/23

[9] Rapid Miner, 2019, https://rapidminer.com/, last accessed 2019/10/23

[10] KNIME End to End Data Science, 2019, https://www.knime.com/, last accessed 2019/10/23

[11] scikit-learn: machine learning in Python, https://scikit-learn.org/stable/, last accessed 2019/10/23

[12] A Short Introduction to the caret Package, https://cran.r-project.org/web/packages/caret/vignettes/caret.html, last accessed 2019/10/23

[13] matplotlib, 2019, https://matplotlib.org/, last accessed 2019/10/23

[14] Tableau, 2019, https://www.tableau.com, last accessed 2019/10/23

[15] Microsoft Power BI, 2019, https://powerbi.microsoft.com, last accessed 2019/10/23

[16] T. Kluyver et al., Jupyter Development Team, “Jupyter Notebooks – a publishing format for reproducible computational workflows”, IOS Press. pp. 87-90, 2016, DOI: 10.3233/978-1-61499-649-1-87

[17] C. Boettiger, “An introduction to Docker for reproducible research, with examples from the R environment”, ACM SIGOPS Oper. Syst. Rev.. 49. 10.1145/2723872.2723882. https://arxiv.org/pdf/1410.0846.pdf

[18] Nature Editors 2012. “Must try harder”. Nature. 483,7391 Mar. 2012, 509–509.

[19] M. D. Wilkinson et al., “The FAIR Guiding Principles for scientific data management and stewardship”, Scientific Data 3, doi : 10.1038/sdata.2016.18, 2016.

[20] E. Cirillo, “TranSMART data exploration and analysis using Python Client and Jupyter Notebook”, 2018, http://blog.thehyve.nl/blog/transmart-data-exploration-and-analysis-using-python-client-and-jupyter-notebook,last accessed 2019/07/22

[21] K. Kerr, “Gartner Named NetApp a Leader in Magic Quadrant for 2019 Primary Storage”, 2019, https://blog.netapp.com/netapp-gartner-magic-quadrant-2019-primary-storage/, last accessed 2019/10/23

[22] OVH SAS. 2018 “Understanding Tier 3 and Tier 4”. https://www.ovh.com/world/dedicated-servers/understanding-t3-t4.xml, last accessed 2018/09/15.

[23] Jupyter Docker Stacks, 2018, https://jupyter-docker-stacks.readthedocs.io/en/latest/, last accessed 2019/10/23

[24] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu and W. Zhou, “A comparative study of Containers and Virtual Machines in Big Data environment”, arXiv:1807.01842v1

 ISBN: 978-1-61208-750-4

[25] L. Hecht, “What the data says about Kubernetes deployment patterns”, 2018, https://thenewstack.io/data-says-kubernetes-deployment-patterns/, last accessed 2019/07/22

[26] Project Jupyter Contributors, “Zero to JupyterHub with Kubernetes”, 2019, https://zero-to-jupyterhub.readthedocs.io/en/latest/, last accessed 2019/07/22

[27] S. Conway, “Survey shows Kubernetes leading as orchestration Platform”, 2018, https://www.cncf.io/blog/2017/06/28/survey-shows-kubernetes-leading-orchestration-platform/, last accessed 2019/07/22

[28] S. Talari, “Why Kubernetes is a great choice for Data Scientists”, https://towardsdatascience.com/why-kubernetes-is-a-great-choice-for-data-scientists-e130603b9b2d, last accessed 2019/07/22

[29] P. Debois and J. Humble, “The DevOps Handbook: how to create World-Class agility, Reliability, and Security in Technology Organizations”, IT Revolution Press, 2016, ISBN: 978-1942788003

[30] NetApp Trident, 2019, https://netapp-trident.readthedocs.io/en/latest/introduction.html, last accessed 2019/10/23

[31] OAuth community site, https://oauth.net/, last accessed 2019/10/23

[32] OpenID Foundation, 2019, https://openid.net, last accessed 2019/10/23

[33] Project Jupyter Team, “Jupyter Enterprise Gateway”, 2016, https://jupyter-enterprise-gateway.readthedocs.io/en/latest/, last accessed 2019/07/22

[34] Apache Airflow Documentation, https://airflow.apache.org/, last accessed 2019/10/23

[35] Pachyderm - Reproducible Data Science that Scales!, 2019, https://www.pachyderm.io/, last accessed 2019/10/23

 ISBN: 978-1-61208-750-4