Nico Herzberg
Christoph Hochreiner
Oliver Kopp
Jörg Lenhard

ZEUS 2018

10^{th} ZEUS Workshop, ZEUS 2018,
Dresden, Germany, 8–9 February 2018
Proceedings

Volume Editors

Nico Herzberg
SAP SE, Ausbildungsleitung SAP Dresden
Postplatz 1, DE-01067 Dresden
nico.herzberg@sap.com

Christoph Hochreiner
TU Wien, Distributed Systems Group
Karlsplatz 13, AT-1040 Wien
c.hochreiner@infosys.tuwien.ac.at

Oliver Kopp
University of Stuttgart, Institute for Parallel and Distributed Systems
Universitätsstraße 38, DE-70569 Stuttgart
oliver.kopp@ipvs.uni-stuttgart.de

Jörg Lenhard
Karlstad University, Department of Mathematics and Computer Science
Universitetsgatan 2, SE-65188 Karlstad
joerg.lenhard@kau.se

Preface

In February 2018, we had the pleasure to organize the 10^{th} edition of the ZEUS Workshop in Dresden, Germany. This workshop series offers young researchers an opportunity to present and discuss early ideas and work in progress as well as to establish contacts among young researchers. For this year's edition, we selected eight regular submissions, two position papers, and one tool demonstration by researchers from Belgium, Egypt, Germany, Italy, and Switzerland for presentation at the workshop. Each submission went through a thorough peer-review process and was assessed by at least three members of the program committee with regard to its relevance and scientific quality. The accepted contributions cover the areas of Microservices, Business Process Management, and the Internet of Things. In addition, the workshop also hosted a tool session to introduce early stage researchers to tools that ease the literature research and the documentation of architectural decisions for software systems.

The workshop program was further enriched by a keynote held by Prof. Dr. Gerhard P. Fettweis on the topic *5G mobile service – first steps of the era after IoT to Tactile Internet*. Furthermore, the workshop participants were invited to visit the Industrial Internet of Things Test Bed at University of Applied Sciences in Dresden, the Blockchain Meetup Saxony at SAP Dresden, and the Volkswagen Transparent Factory. The best presentation award was given to Justus Bogner from the Herman Hollerith Center Böblingen, for his presentation of the paper *Analyzing the Relevance of SOA Patterns for Microservice-Based Systems*.

The workshop was generously sponsored by SAP SE.

Dresden, February 2018

Nico Herzberg
Christoph Hochreiner
Jörg Lenhard
Oliver Kopp

Organization

Steering Committee

Oliver Kopp	University of Stuttgart
Jörg Lenhard	Karlstad University
Christoph Hochreiner	TU Wien

Local Organizer

Nico Herzberg	SAP SE

Program Committee Chairs

Nico Herzberg	SAP SE
Christoph Hochreiner	TU Wien
Oliver Kopp	University of Stuttgart
Jörg Lenhard	Karlstad University

Program Committee

Saimir Bala	Vienna University of Economics and Business
Felix W. Baumann	TWT GmbH Science & Innovation
Domenico Bianculli	University of Luxembourg
Daniele Bonetta	Oracle Labs
Jürgen Cito	University of Zurich
Vincenzo Ferme	USI Lugano
Alessio Gambi	Saarland University
Matthias Geiger	University of Bamberg
Georg Grossmann	University of South Australia
Simon Harrer	University of Bamberg
Thomas Heinze	University of Jena
Pascal Hirmer	University of Stuttgart
Christoph Hochreiner	TU Wien
Conrad Indiono	University of Vienna
Meiko Jensen	ULD Schleswig-Holstein
Stefan Kolb	University of Bamberg
Oliver Kopp	University of Stuttgart
Jörg Lenhard	Karlstad University
Daniel Lübke	Leibniz Universität Hannover
Matteo Nardelli	University of Rome Tor Vergata
Jan Sürmerli	Humboldt University of Berlin

Subreviewers

Robin Lichtenthäler	University of Bamberg
Stefan Winzinger	University of Bamberg

Sponsoring Institutions

SAP Deutschland AG

Table of Contents

Challenges of Microservices Architecture: A Survey on the State of the Practice

Javad Ghofrani and Daniel Lübke

Leibniz Universtität Hannover, Hannover, Germany,
{javad.ghofrani,daniel-luebke}@inf.uni-hannover.de

Abstract. Microservices have been one of the fastest-rising trends in the development of enterprise applications and enterprise application landscapes. Even though various mapping studies investigated the open challenges around microservices from literature, it is difficult to have a clear view of existing challenges in designing, developing, and maintaining systems based on microservices architecture as it is perceived by practitioners. In this paper, we present the results of an empirical survey to assess the current state of practice and collect challenges in microservices architecture. Therefore, we synthesize the 25 collected results and produce a clear overview for answering our research questions. The result of our study can be a basis for planning future research and applications of microservices architecture.

Keywords: Empirical software engineering, State of Practice, Microservices, Software Architecture

1 Introduction

Microservices architecture (MSA) is based on a share-nothing philosophy and relies on a long evolving experience in software engineering and system design. This architectural style structures a system as a set of loosely-coupled small services which are isolated in small coherent and autonomous units [9].

Many organizations, such as Amazon, Netflix, and the Guardian, utilize MSA to develop their continuous delivery of large and complex applications while providing flexibility and diversity of technology stack [2]. In addition to structuring the development of the systems, design principles of MSA are used to migrate systems with traditional architectural style into MSA. Since MSA is still young, there are many open issues and optimization possibilities in this field. Even though many research papers (e.g., Vural et al. [8]) tried to identify the issues and proposed solutions from literature around MSA, many aspects of the practical challenges in MSA are still unexplored. This makes it difficult for researchers to have a realistic overview of the real challenges encountered in practice related to MSA and their potential for academic and industrial adoption [2]. The goal of this paper is to characterize the current state of the practice about MSA and provide a complementary overview on the existing scientific research. In order to achieve this goal, we conducted an online survey. Specifically, we

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

selected three research questions related to the existing challenges and solutions from MSA in the practice. **RQ1**: *What are the main challenges/concerns in the design and the development process of microservices?* **RQ2**: *What are the main reasons leveraging and preventing the usage of systematic approaches in microservices architectures?* **RQ3**: *Are there any suggestions or solutions from the experts to improve aspects of the microservices architecture?*

The main contributions of this paper are: (i) providing an up-to-date map of the state of the practice in MSA and its complexities for future research; (ii) an evaluation of the potential research field on MSA. The audience of this paper are both researchers interested to supplement and update their overview of most recent challenges in this field for their future contributions as well as practitioners interested to get an overview of existing challenges and solutions, thereby making better decisions for their organization.

The rest of the paper is organized as follows. Section 2 gives an overview on the existing reviews from literature, whereas the design of our study is presented in Section 3. We elaborate the results of our study in Section 4 by putting them in a broader perspective for answering the proposed research questions. With Section 5 we conclude the paper and discuss the future work. The full survey data including the questionnaire and preprocessed results is shared and available online [5].

2 Related Work

As described in the previous section, the existing academic studies reviewed the literature to provide an overview of open challenges in MSA. There is no academic survey on the state of the practice in MSA as of now. As a basis for our study, we consider the following systematic literature reviews and mapping studies conducted about state of the art in MSA.

Pahl and Jamshidi [7], Alshuqayran et al. [1], and Di Francesco et al. [2] conducted systematic mapping studies about MSA. Pahl and Jamshidi [7] investigated 21 publications until 2015 and reported the existing research trends and directions in this field with regards to applications in the cloud. The informal survey of Dragoni et al. [3] around MSA opens an academic viewpoint to the open problems for novices. The main difference between our study and [1–3, 7] is the research approach, which is used to provide an overview of open issues and challenges in MSA. The previous approaches prefer the academical aspect of existing literature while we explore the practical side of the domain. The provided results by Pahl and Jamshidi [7] are subject to the passage of time and are perhaps even outdated because of the fast progress of MSA's underlying technologies.

3 Study Design

Since surveys provide a better overview of usage of technology in the software industry [4], we performed an online survey to get the practical picture of MSA.

We designed our survey in a brain storming session with a group of five researchers and created a mind-map. We created a list of questions and possible answers based on our mind-map. An iterative review process is performed to assure that the survey questions are comprehensive and relevant to the domain. Finally, we performed pilot tests with two other volunteers.

We created our online survey based on the guidelines described by Jacob et al. [6]. Our survey is mainly based on using multiple choice and numerical questions to provide an "other" option to consider missing options. The answers of "other" options are used either to create new categories or considered as belonging to one of the existing categories by analyzing the survey results. Self-containment of the survey questions are also adhered by providing required information about the context or expressions of a question. In order to avoid long questionnaire for the intended audience, we split the questionnaire into five parts: Profiling, Domain, Modeling, Non-functional, and a Final part. The participants were informed about the results of the survey after leaving their email addresses in the final step of the survey. Our survey was available on the LimeSurvey Server, hosted by Software Engineering Group of Leibniz Universität Hannover, between November 17, 2017 and January 5, 2018.

First, we reached the potential participants via our personal contacts and asked them to take part and also forwarded our invitations to potential participants that they may know. In addition, we posted the survey invitation to several online communities, XING, meet-up groups, and advertised the survey via mailing lists of the NGINX, Inc. In order to control who was answering our survey, we started our survey with a filter question about taking part in a project with relevance to MSA.

4 Results

In total, 40 experts participated in our survey and 25 of them has answered *yes* to the filter question about being involved in any project or program that are related to the microservices architecture. Since our research is related to the practical part of the MSA, we consider only the answers of the participants who answered *yes* to the filter question and answered at least one of the other questions from our survey. The Majority of the respondents, 19 (76%), are practitioners from industry whereas only 2 of them (0.08%) are from academia. Furthermore, 4 participants are active in both academic and industrial contexts. By asking about the role of the participants in their current activities, 17 (68%) stated that they are developers, 11 (44%) system architects, and 3 (12%) of them are team leaders. All of the team leaders and 8 of the system architects are developers simultaneously. Relative to experience with MSA, 11 (44%), have between 1 and 2 years, and 8 (32%) have between 2 and 5 years of experience. 21 participants stated that they use agile methods in their organizations.

4.1 RQ1: What are the main challenges/concerns in the design and the development process of microservices?

To answer RQ1, we asked our participants about their main challenges in the development process of microservices. According to the 8 comments and free-text answers of the survey to this question, the distributed nature of the MSA is one of the main challenges in the development and debugging of the systems based on this architectural style. "Too many repositories to maintain", "Hard to find issues in a distributed system. Rarely any benefit", "networking between dockers", "sharing base data among different services considering performance limitations", and "debugging a microservice that relies on other services can be tricky" are some of these comments. Next common challenge is the skill and knowledge. It seems that there are difficulties between customers and development teams in "understanding MS of the Top Management/Director levels", "getting the right developers/engineers", and "Changing the people's minds that are used to traditional monoliths". Finally, "correct separation of domains" and "finding the appropriate service cuts" are the third majority of the concerns stated by our participants. Nevertheless, they have also recommended to use MSA instead of other architectural approaches in case of "Distributed teams that can have logical separations".

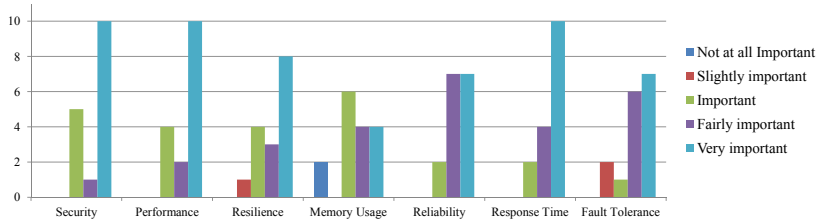

Fig. 1. Priority of MSA features that should be optimized, according to the number of votes given by practitioners

In order to get an overview of the main concerns of our participants regarding nonfunctional features of their MSA, we asked "How important is it for you to optimize the following features of your microservice architecture?"

- Security
- Performance
- Resilience
- Memory Usage
- Reliability
- Response time
- Fault tolerance [1]

[1] We use 5 levels of importance: Not at all important, Slightly important, Important, Fairly Important, and Very important

Fig. 1 depicts the frequency and the importance level of each the features. One can see that Security, Performance, and Response Time are mentioned as very important to optimize in MSA. The optimization of Resilience, Reliability and Fault Tolerance are considered to be the next priorities while the Memory Usage is only important.

Furthermore, we asked our participants " What were your goals when deciding on a microservice architecture?". Fig. 2 shows that Scalability (16/25) and Agility (13/25) are commonly desired compared to Extendability (9/25), Maintainability (9/25), and Reliability (7/25).

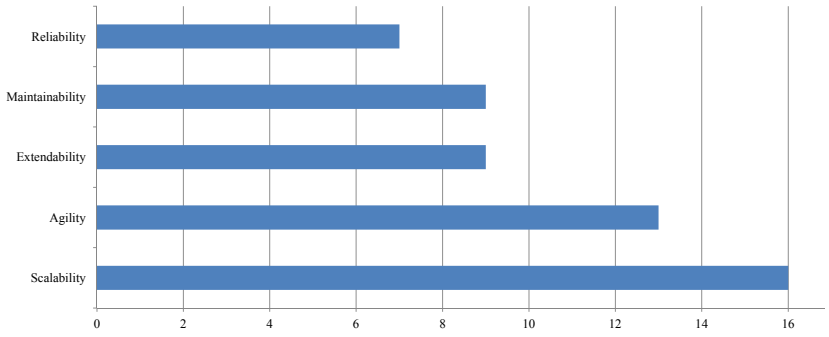

Fig. 2. Participants' primary intention for using a microservice architecture

4.2 RQ2: What are the main reasons leveraging and preventing the usage of systematic approaches in microservices architectures?

We asked the participants "How do you derive service boundaries?". On the one hand, a big amount of the participants (7/25) use Manually / Good Feeling methods based on their experiences and skills. Looking at the skills of these participants reveals that most of them have less than 2 years of experience with MSA. On the other hand, 6/25 of participants inform us that they use systematic approaches, especially Domain Driven Design, for deriving the service boundaries. It shows a significant correlation (85%) with their experience on MSA, between 2 to 5 years. Automatic methods, such as tooling, formal methods, and algorithms for deriving service boundaries were used by 3 participants. Two of them have less than 2 years of experience in MSA and only one has between 2 and 5 years of experience. These answers are depicted in Fig. 3.

The next question was: "Which notation(s) do you use to describe your architecture?". As shown in Fig. 4, 8/25 of the participants stated that they use Graphical Modeling Languages (GML) because of agility, simplicity, and easily available GML tools. 2/25 use Textual Modeling Languages (TML) and

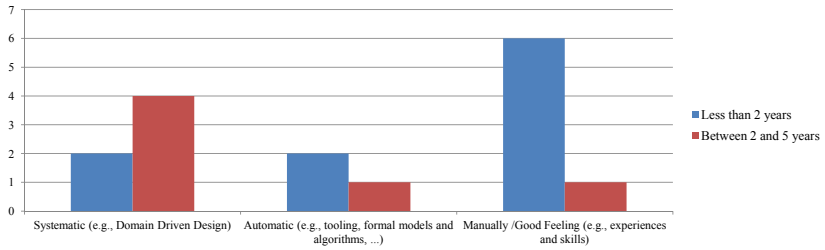

Fig. 3. Popularity of each method among the participants with less than 2 years (blue bars) and between 2 to 5 years (red bars) of MSA experience

4/25 of them use Domain Specific Languages (DSL). Nevertheless, 9/25 of participants stated that they do not use any notation in this context. None of the participants named any tool, technique, and framework for modeling their MSA which indicates the limited awareness on existing tools for MSA among the practitioners.

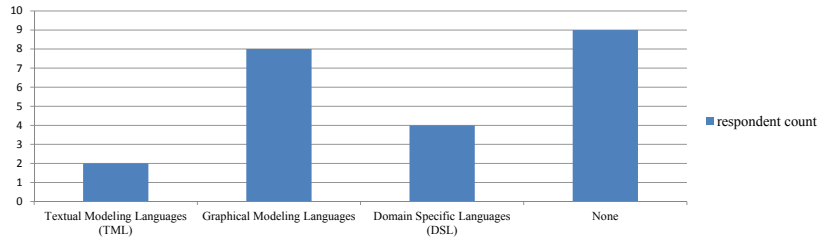

Fig. 4. Notations that are used by practitioners to describe their MSA, according to 25 collected responses

4.3 RQ3: Are there any suggestions or solutions from the experts to improve aspects of the microservices architecture?

Finding a trade-off between reusing and developing the architectural artifacts is a well-known issue in any software architecture. We asked the participants about their opinion on usage of artifacts from third-parties within a MSA as the following question: "If you use any artifacts from third-parties in your microservice architecture, do you consider the following features? How important are they for you?".

- Last Release Date
- Security

– Last Release
– Memory Usage
– License [2]

As Fig. 5 illustrates, Security, License, and Memory usage are the most respected concerns according to the given priorities. Furthermore, one respondent mentioned that "he/she often finds out about memory footprint or computing time only after he/she has integrated and tested" about third-party frameworks or libraries. There is also an interesting suggestion from a respondent to start renaming microservice nomenclature in order to facilitate the utilization of Domain Driven Design tools within the MSA context.

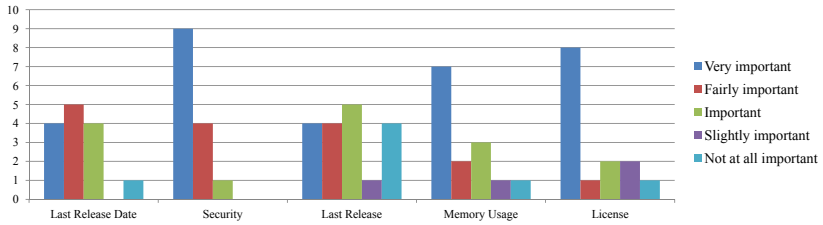

Fig. 5. The importance of various aspects of artifacts from third-parties according to the collected responses

5 Conclusions and Future Work

In this paper we reported the results of our online survey that we conducted among experts of Microservices architecture (MSA). We have investigated survey responses to answer our three research questions about challenges in MSA. Since the majority of our experts are from industry, our research reflects the practical issues in MSA. Our work can be used as a complementary work to the existing literature reviews to guide researchers to the open issues and problems in MSA and offer an overview from practical point of view.

Among the collected responses, the lack of notations, methods, and frameworks to architect MSA can be considered as important points. The lack of tool or framework support for selecting third-party artifacts according to their features, e.g. security, last release, as well as the shortage of knowledge of the practitioners about systematic methods are the top main gaps, which can be addressed in the future work. According to the results of our survey, optimization in Security, Response Time, and Performance have higher priorities than Resilience, Reliability, Fault Tolerance, and Memory Usage.

[2] We use 5 levels of importance: Not at all important, Slightly important, Important, Fairly Important, and Very important

Future work includes (i) conducting a survey each year to monitor the change over time, (ii) performing a new survey with a larger scope to collect more details about further aspects of MSA in the practice and (iii) proposing solutions for the identified gaps.

References

[1] Alshuqayran, N., Ali, N., Evans, R.: A systematic mapping study in microservice architecture. In: Service-Oriented Computing and Applications (SOCA), 2016 IEEE 9th International Conference on. pp. 44–51. IEEE (2016)
[2] Di Francesco, P., Malavolta, I., Lago, P.: Research on architecting microservices: Trends, focus, and potential for industrial adoption. In: Software Architecture (ICSA), 2017 IEEE International Conference on. pp. 21–30. IEEE (2017)
[3] Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: yesterday, today, and tomorrow. In: Present and Ulterior Software Engineering, pp. 195–216. Springer (2017)
[4] Fink, A.: The survey handbook, vol. 1. Sage (2003)
[5] Ghofrani, J., Lübke, D.: Online material for survey on challenges of microservices architecture (2018), https://doi.org/10.6084/m9.figshare.5852598
[6] Jacob, R., Heinz, A., Décieux, J.P.: Umfrage: Einführung in die Methoden der Umfrageforschung. Walter de Gruyter (2013)
[7] Pahl, C., Jamshidi, P.: Microservices: A systematic mapping study. In: CLOSER (1). pp. 137–146 (2016)
[8] Vural, H., Koyuncu, M., Guney, S.: A Systematic Literature Review on Microservices, pp. 203–217. Springer International Publishing, Cham (2017)
[9] Wolff, E.: Microservices: Flexible Software Architecture. Addison-Wesley Professional (2016)

Analyzing the Relevance of SOA Patterns for Microservice-Based Systems

Justus Bogner[1,2], Alfred Zimmermann[1], and Stefan Wagner[2]

[1] Reutlingen University of Applied Sciences, Germany
{justus.bogner,alfred.zimmermann}@reutlingen-university.de
[2] University of Stuttgart, Germany
{justus.bogner,stefan.wagner}@informatik.uni-stuttgart.de

Abstract. To bring a pattern-based perspective to the SOA vs. Microservices discussion, we qualitatively analyzed a total of 118 SOA patterns from 2 popular catalogs for their (partial) applicability to Microservices. Patterns had to hold up to 5 derived Microservices principles to be applicable. 74 patterns (63%) were categorized as fully applicable, 30 (25%) as partially applicable, and 14 (12%) as not applicable. Most frequently violated Microservices characteristics were *Decentralization* and *Single System*. The findings suggest that Microservices and SOA share a large set of architectural principles and solutions in the general space of Service-Based Systems while only having a small set of differences in specific areas.

Keywords: Microservices, SOA, Service-Based Systems, Design Patterns

1 Introduction

Over the last decade, Service-Oriented Computing (SOC) [13] established itself as one of the most important paradigms for distributed systems. The implementation of enterprise-wide software landscapes in the style of Service-Oriented Architecture (SOA) [3] brought benefits with respect to encapsulation, interoperability, composition, reuse, loose coupling, and maintainability. However, increased standardization and governance efforts, higher architectural and technological complexity, and sometimes vendor or product lock-in caused frustration and lead to failed SOA adoption projects [11]. In recent years, Microservices [7,12] as an agile, DevOps-focused, decentralized service-oriented variant with fine-grained services quickly gained in popularity and tried to address some of the issues with both large monolithic applications as well as "traditional" Service-Oriented Systems based on SOAP/WSDL and a central Enterprise Service Bus (ESB).

There is still an ongoing discussion in industry and academia about the differentiation of SOA and Microservices. Some see it as a very new architectural style that needs to be treated very differently ("revolutionary" perspective), some see it merely as a specialization of SOA, e.g. "fine-grained SOA" ("evolutionary" perspective). While many papers have been published on the subject, so far

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

no comprehensive pattern-based approach to compare the two has been taken. Software design patterns are a well-established form to document proven solutions to recurring design problems within a specific context in a technology-agnostic yet easily implementable way. They base their origin in Alexander's building pattern language [1] and went mainstream with the famous Gang of Four "Design Patterns" [8]. There is large catalog of documented service-oriented patterns that emerged over the years as a result of growing SOA industry experience. However, it is not fully clear, if these patterns are of value for Microservice-Based Systems.

The contribution of this work is to add a pattern-based perspective to the "SOA vs. Microservices" discussion by analyzing the applicability of existing SOA patterns for a Microservices context. To create a basis for important principles of Microservice-Based Systems, Section 2 introduces existing comparisons of the two service-based architectural styles. Section 3 outlines the detailed scope and research method of the pattern-based approach, while Section 4 presents the results. Finally, Section 5 closes with a summary, limitations, and an outlook on potential follow-up research.

2 Related Work: SOA vs. Microservices

Several perspectives on the comparison of SOA and Microservices have been published so far. Zimmermann first compares the two most popular definitions of Microservices, namely the one of Lewis/Fowler and the definition of Newman [19]. He distills common tenets and warns that the two definitions mix concerns related to process, organization, architecture, and development, which should be avoided when defining an architectural style. He then analyzes the identified tenets for SOA pendants and finds similarities for most of them. For him, the largest differences show with respect to decentralized governance, infrastructure automation, independently deployable services, and lightweight communication as opposed to a central ESB. Based on these findings, the analysis concludes that Microservices can be seen as a specific development and deployment approach for SOA.

Similarly, Dragoni et al. come to the conclusion that Microservices are "the second iteration" of the SOC and SOA concepts with the aim to strip away complexity and to focus on the development of simple and lightweight services [2]. While SOA addresses the enterprise workflow level, Microservices aim for a smaller application-level scope. Other apparent differences for them include independent bounded contexts with small services, the high degree of automation, the organizational aspects related to DevOps teams ("you build it, you run it"), the preference of choreography over orchestration, and a potentially higher degree of technological heterogeneity.

Xiao et al. describe Microservices and SOA as allies that should be leveraged to enable a bi-modal or two-speed IT in the digital age [18]. Their comparison highlights autonomy, size, and the development and deployment cycle as main differences. Additionally, decentralized governance and different communication and message exchange protocols are pointed out. Apart from these, a lot of

similarities are mentioned, especially the focus on organizing services around business capabilities and service-oriented principles like statelessness, reuse, and abstraction.

Salah et al. take a broad evolutionary perspective and compare the client/server architecture, mobile agents architecture, SOA, and Microservices [17]. They describe a direct line of evolution from client/server to SOA and from there on further down to Microservices. Main differences for them include high service independence and decentralization, fine-grained services with bounded contexts, fast software delivery, and lightweight communication via "dumb pipes" and no middleware focus.

Lastly, the most "revolutionary" perspective is taken by Richards [14]. He argues that SOA focuses on large, complex, enterprise-wide systems whereas Microservices target small to medium web-based applications. Likewise, SOA follows a "share-as-much-as-possible" approach while Microservices are based on the "share-as-little-as-possible" principle. For Richards, Microservices are located on the other side of the service-oriented spectrum as SOA. He focuses very much on differences, not so much on commonalities. Other notable Microservices differences presented are the low degree of centralization and standardization, a very small number of lightweight communication protocols as opposed to SOA's protocol-agnostic heterogeneous interoperability provided by an ESB, and the focus on bounded contexts as opposed to SOA's focus on abstraction and business functionality reuse.

To the best of our knowledge, there are currently no publications solely on the topic of SOA patterns and Microservices. There are some publications concerning patterns specifically tailored for Microservices [9, 10, 15], but the authors do not really explain the relation to SOA. Moreover, several of these Microservices patterns are not new and have been used in other contexts before (including SOA). A detailed analysis would be interesting, but goes beyond the scope of this paper.

3 Scope and Research Method

All comparisons in Section 2 focus on design characteristics, principles, or applied technologies. A different approach would be to analyze existing SOA patterns for applicability to a Microservices context. This architecture- and design-centric perspective has the positive side-effect of providing a list of candidate patterns potentially usable in a Microservice-Based System. We use Erl's [4, 5] and Rotem-Gal-Oz's [16] books as sources for SOA patterns, as they are well established in industry and academia and have minimal overlap. From literature (including the publications in Section 2) we compiled the following list of Microservice-specific principles.

- **Bounded Context:** fine-grained services according to Bounded Contexts [6]
- **Decentralization:** decentralization of control and management, low degree of standardization, choreography over orchestration

- **Lightweight Communication:** communication via RESTful APIs or lightweight messaging (no ESB, no workflow engine, etc.), "dumb pipes"
- **Single System:** building a single Service-Based System of medium size
- **Technological Heterogeneity:** support of diverse programming languages, databases, or used frameworks/libraries

With these criteria, we qualitatively analyzed the patterns in our 3 sources and documented if the usage of a pattern violates the compiled characteristics. Based on the violations, a pattern is categorized as **fully applicable**, **partially applicable** (with certain limitations/modifications), or **not applicable**. Some examples: The pattern *Enterprise Inventory* that provides architecture, standardization, and governance boundaries for every service within the enterprise is categorized as *not applicable*, because this violates the Microservice principles *Decentralization*, *Single System*, and *Technological Heterogeneity*. Likewise, the pattern *Protocol Bridging* that enables communication between consumers and providers that rely on different protocols is rated *partially applicable*, because this is usually not necessary in a Microservice-Based System. The principles *Lightweight Communication* and *Single System* could be violated by this. However, there could be rare evolutionary use cases where this pattern could indeed be applied, e.g. when a service that uses message-based communication (e.g. AMQP) should interact with one that relies on RESTful HTTP because of new or changed requirements. Lastly, the pattern *Lightweight Endpoint* where a series of fine-grained capabilities replaces a single coarse-grained capability to avoid wasteful data exchange and consumer-side processing is categorized as *fully applicable*. It violates none of the defined Microservices principles and is in fact in line with the core values of this architectural style.

The aggregated results of all these pattern categorizations are then used for further analysis and to provide answers to the following research questions:

RQ1: To what degree are SOA design patterns applicable to Microservices?

RQ2: What pattern categories are the most or least applicable?

RQ3: What are the most frequent Microservice-specific properties violated by not or partially applicable patterns?

4 Results: Applicability of SOA Patterns to Microservices

Erl's catalog comprises a total of 92 patterns (85 SOA patterns [4] and 7 REST-inspired patterns [5]). These patterns are structured into 5 different categories, namely *Service Inventory Design Patterns* (24 patterns), *Service Design Patterns* (31 patterns), *Service Composition Design Patterns* (23 patterns), *Compound Design Patterns* (7 patterns), and *REST-Inspired Patterns* (7 patterns). Rotem-Gal-Oz's more compact book presents 26 patterns [16] in 6 categories, namely *Foundation Structural Patterns* (5 patterns), *Performance, Scalability, and Availability Patterns* (6 patterns), *Security and Manageability Patterns* (5 patterns), *Message Exchange Patterns* (4 patterns), *Service Consumer Patterns* (3 patterns), and *Service Integration Patterns* (3 patterns). So all in all, we

analyzed 118 SOA patterns with very few duplicates (examples being *Service Bus* or *Orchestration*).[3]

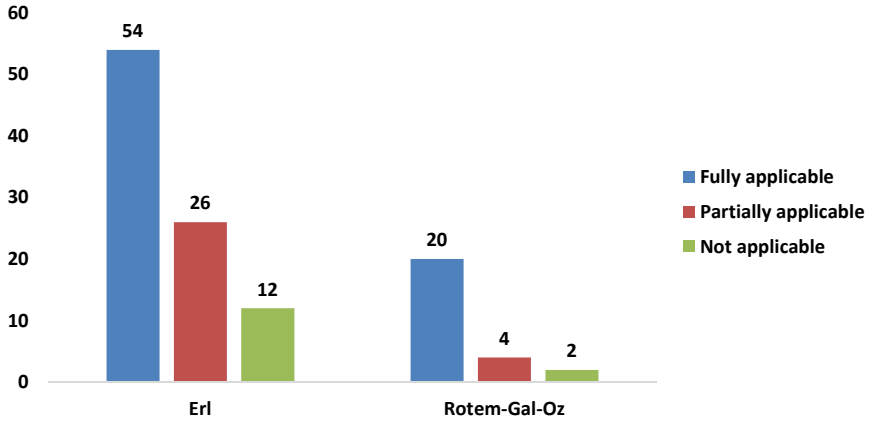

Fig. 1. SOA Pattern Applicability to Microservices

From Erl's 92 patterns, 54 (59%) patterns were found to be **fully applicable**, 26 (28%) to be **partially applicable** and only 12 (13%) were categorized as **not applicable**. That means that 87% of the patterns were estimated at least partially applicable in a Microservices context. For Rotem-Gal-Oz's smaller catalog, the numbers were even higher: Of the 26 patterns, 20 (77%) were categorized as **fully applicable**, 4 (15%) as **partially applicable**, and only 2 (8%) were deemed **not applicable**. So 92% of these patterns were found to be at least partially applicable for Microservices. When combining both catalogs (118 patterns), this accounts for 74 (63%) **fully applicable**, 30 (25%) **partially applicable**, and 14 (12%) **not applicable** patterns (see Fig. 1).

When looking at Erl's 5 pattern categories, an immediate observation is that all of the 7 *REST-Inspired Patterns* are fully applicable, which seems understandable in light of Microservice-related communication preferences. Furthermore, 22 of 31 *Service Design Patterns* (71%) and 15 of 23 *Service Composition Design Patterns* (65%) were fully applicable, which makes these two categories also very useful design sources when building Microservice-Based Systems. The 7 *Compound Design Patterns* were the least applicable category with 0 fully and only 4 partially applicable patterns (57%). This can be explained with the complexity and centralized nature of these patterns. For Rotem-Gal-Oz's 6 categories, both *Message Exchange Patterns* and *Service Consumer Patterns* are 100% fully applicable. Moreover, *Performance, Scalability, and Availability Patterns* with 83% and *Security and Manageability Patterns* with 80% fully applicable patterns

[3] For details see: https://github.com/xJREB/research-soa-patterns-for-microservices

are mentionable. All in all, the categories here consisted of fewer patterns, which makes it harder to compare them with Erl's.

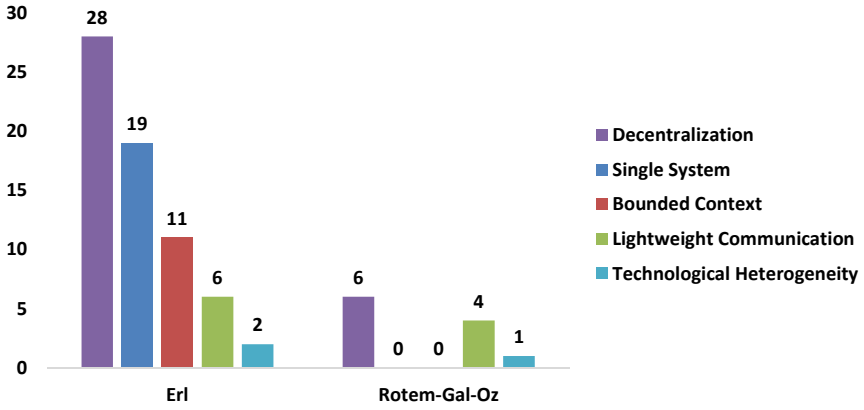

Fig. 2. Violated Microservice-related Principles of Not Fully Applicable SOA Patterns

When analyzing the most frequent causes why patterns were not or only partially applicable (see Fig. 2), the **Decentralization** characteristic was the most violated one (42% of all violations for Erl, 55% for Rotem-Gal-Oz). After that, **Single System** (29%) and **Bounded Context** (17%) were the next frequent violations in Erl's catalog, with **Lightweight Communication** only accounting for 9%. Interestingly, **Technological Heterogeneity** was violated only twice by Erl's patterns (*Enterprise Inventory* and *Domain Inventory*) and in Rotem-Gal-Oz's catalog only by one single pattern (*Service Host*). This can be explained with the technology-agnostic form of design patterns. Moreover, SOA systems are no strangers to diversity, so even an increase in technological heterogeneity in a Microservice-Based System will not invalidate the vast majority of patterns.

5 Summary and Conclusion

Based on 5 derived Microservices principles, we qualitatively analyzed the applicability of SOA design patterns for the context of Microservices. Of 118 patterns, 74 (63%) were found to be **fully applicable**, 30 (25%) **partially applicable**, and 14 (12%) **not applicable**. The most violated principles were *Decentralization* and *Single System* while *Technological Heterogeneity* had very little impact. The findings suggest that from a pattern-based perspective, Microservices and SOA have some small distinct areas of differences, but share a large set of design-related commonalities.

However, since descriptions of Microservices (unlike descriptions of "pure" architectural styles) cover other areas than architecture and design (e.g. process, organization, development, operations, etc.), differences in these other areas may be much more apparent. Since a definition of SOA as an architectural style should not define or restrict these areas, this seems to support the view of [19] that Microservices can be seen as a specific development and deployment approach for SOA.

Limitations of our work are the qualitative nature of the comparison. Results precision could have benefited greatly from a more rigorous method to rate patterns and state that a principle was violated. Similarly, an external validation of the pattern ratings by experts in the field of Microservices would have improved the results further. This would have reduced the possibility of subjective bias and increased the reproducibility of the study. Follow-up research could include such methods as well as trying to identify SOA patterns in existing Microservice-Based Systems. Lastly, it will be interesting to analyze the currently forming catalog of Microservices patterns to check for either "SOA backwards compatibility" or existing SOA pattern ancestors.

Acknowledgments This research was partially funded by the Ministry of Science of Baden-Württemberg, Germany, for the Doctoral Program "Services Computing" (http://www.services-computing.de/?lang=en).

References

1. Alexander, C., Ishikawa, S., Silverstein, M., i Ramió, J.R., Jacobson, M., Fiksdahl-King, I.: A Pattern Language. Gustavo Gili (1977)
2. Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: Yesterday, Today, and Tomorrow. In: Present and Ulterior Software Engineering, pp. 195–216. Springer International Publishing, Cham (2017)
3. Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, Upper Saddle River, NJ, USA (2005)
4. Erl, T.: SOA Design Patterns. Pearson Education, Boston, MA, USA (2009)
5. Erl, T., Carlyle, B., Pautasso, C., Balasubramanian, R.: SOA with REST: Principles, Patterns & Constraints for Building Enterprise Solutions with REST. The Prentice Hall Service Technology Series from Thomas Erl, Pearson Education (2012)
6. Evans, E.: Domain-driven Design: Tackling Complexity in the Heart of Software. Addison-Wesley (2004)
7. Fowler, M.: Microservices Resource Guide (2015), `http://martinfowler.com/microservices`
8. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Boston, MA, USA (1994)
9. Gupta, A.: Microservice Design Patterns (2015), `http://blog.arungupta.me/microservice-design-patterns`
10. Krause, L.: Microservices: Patterns and Applications. Lucas Krause (2015)
11. MacLennan, E., Van Belle, J.P.: Factors affecting the organizational adoption of service-oriented architecture (SOA). Information Systems and e-Business Management 12(1), 71–100 (2014)

12. Newman, S.: Building Microservices: Designing Fine-Grained Systems. O'Reilly Media, 1st edn. (2015)
13. Papazoglou, M.P.: Service-oriented computing: concepts, characteristics and directions. In: Proceedings of the 7th International Conference on Properties and Applications of Dielectric Materials (Cat. No.03CH37417). pp. 3–12. IEEE Comput. Soc (2003)
14. Richards, M.: Microservices vs. Service-Oriented Architecture. O'Reilly Media, Sebastopol, CA (2016)
15. Richardson, C.: Microservices Patterns. Manning Publications (2018)
16. Rotem-Gal-Oz, A.: SOA Patterns. Manning, Shelter Island, NY (2012)
17. Salah, T., Jamal Zemerly, M., Chan Yeob Yeun, Al-Qutayri, M., Al-Hammadi, Y.: The evolution of distributed systems towards microservices architecture. In: 2016 11th International Conference for Internet Technology and Secured Transactions (ICITST). pp. 318–325. IEEE (2016)
18. Xiao, Z., Wijegunaratne, I., Qiang, X.: Reflections on SOA and Microservices. In: 2016 4th International Conference on Enterprise Systems (ES). pp. 60–67. IEEE (2016)
19. Zimmermann, O.: Microservices tenets. Computer Science - Research and Development 32(3-4), 301–310 (2017)

Mutation Testing for Microservices

Stefan Winzinger

Distributed System Group, University of Bamberg, Germany
stefan.winzinger@uni-bamberg.de

Abstract. The microservice architectural style is currently of great interest both to research and industry. Since applications built by this style consist of many loosely coupled services, it is necessary to test their interactions by test suites. A crucial question is to decide which test cases are necessary and able to detect errors. A method to assure the quality of test cases is mutation testing. However, there are no mutation operators available yet which would enable the application of mutation testing specifically for microservices. This paper presents preliminary ideas for the creation of possible mutation operator whose application could help assure the quality of test cases by using mutation testing and therefore improve the quality of microservice systems.

Keywords: microservices, mutation testing, mutation operator

1 Motivation

Microservices have emerged as a trend over the last years and can be defined as small, autonomous services that work together [9]. The independence of the components in a microservice architecture makes it possible to test them in isolation. But testing on a higher level can become very difficult, especially for larger systems with many connections between the services [7]. Therefore, test cases are needed that detect faults, which only emerge while using several services in combination, as these faults are not detected while testing a single service.

A method to evaluate the potential of a test case suite is mutation testing. In general, mutation testing is a fault-based testing technique which creates a faulty set of programs by seeding faults, which are often done by programmers, into the program. A faulty program is called a mutant. By running the test suite against each of the mutants, a mutant is "killed" as soon as its fault is detected. The 'mutation score' is the ratio of the detected faults over the total number of seeded faults [8]. It improves with every mutant killed.

By mutating the program many faults can be produced at low cost [6]. However, mutation operators providing the rules to create faults are required in order to create faulty programs. Using mutation operators can produce programs whose faults are similar to those of real programs [2]. Therefore, mutation operators for microservices could help to create mutants automatically which would facilitate the assessment of the test case quality. Thus, quality and speed of test execution could be improved resulting in a faster delivery for the customer.

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

Mutation testing can be applied at unit level, integration level and specification level [8]. Since there are no mutation operators focusing on the microservice architecture identified yet, we want to identify some mutation operators being useful for microservices and evaluate them by applying them on a running system.

2 Research Outline

For our future work, we plan to investigate several generic mutation operators on unit, integration and specification level. We will consider existing mutation operators and their suitability for microservices and define new mutation operators adapted to the microservice architectural style to apply them to a technology used in practice:

Unit level
There are already many mutation operators defined for specific programming languages (e.g. C, Java or C# [9]). E.g. considering C, these mutation operators change a statement, an operator, a variable or a constant [1]. These mutation operators will probably be applicable for testing single microservices depending on the language used. However, faults introduced by using these mutation operators are not characteristic for microservices and can be detected by testing services in isolation. Therefore, no microservice-specific mutation operators can be introduced at unit level but established mutation operators can be used to assess test suites for isolated microservices.

Integration level
Mutation operators at the integration level are more interesting since microservices can be interpreted as independent units which communicate by using interfaces. Therefore, the integration level is a more promising area to define characteristic mutation operators. As described in [4] or [5], interfaces can be changed by removing or modifying parameters. This is a promising approach, especially since a potential failure of a service can be simulated by using this approach. Additionally, it would be possible to add a delay to the delivery of messages in order to simulate a network congestion which would force an alternative service to handle this message.

Specification level
In [3] Estelle Specifications are used to specify a system. Estelle Specifications are a Formal Description Technique which describes a system hierarchically. Some mutation operators are introduced which focus on the structure of the system. Especially mutation operators modifying the control flow among components could be transferred to microservices by considering the network of services. By changing the control flow of services (e.g. parallel instead of sequential execution of services and vice versa) faults might occur which should be covered by test cases.

Finally, the most promising mutation operators shall be applied to several test case suites on a running system and be evaluated regarding their suitability to assess the quality of test cases.

References

1. Agrawal, H., DeMillo, R., Hathaway, R., Hsu, W., Hsu, W., Krauser, E., Martin, R.J., Mathur, A., Spafford, E.: Design of mutant operators for the c programming language. Tech. rep., Technical Report SERC-TR-41-P, Software Engineering Research Center, Department of Computer Science, Purdue University, Indiana (1989)
2. Andrews, J.H., Briand, L.C., Labiche, Y.: Is mutation an appropriate tool for testing experiments? In: Proceedings of the 27th international conference on Software engineering. pp. 402–411. ACM (2005)
3. De Souza, S.D.R.S., Maldonado, J.C., Fabbri, S.C.P.F., De Souza, W.L.: Mutation testing applied to estelle specifications. In: System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on. pp. 10–pp. IEEE (2000)
4. Delamaro, M.E., Maldonado, J.C., Mathur, A.P.: Integration testing using interface mutation. In: Software Reliability Engineering, 1996. Proceedings., Seventh International Symposium on. pp. 112–121. IEEE (1996)
5. Delamaro, M.E., Maldonado, J.C., Pasquini, A., Mathur, A.P.: Interface mutation test adequacy criterion: An empirical evaluation. Empirical Software Engineering 6(2), 111–142 (2001)
6. Do, H., Elbaum, S., Rothermel, G.: Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Software Engineering 10(4), 405–435 (2005)
7. Dragoni, N., Giallorenzo, S., Lafuente, A.L., Mazzara, M., Montesi, F., Mustafin, R., Safina, L.: Microservices: yesterday, today, and tomorrow. In: Present and Ulterior Software Engineering, pp. 195–216. Springer (2017)
8. Jia, Y., Harman, M.: An analysis and survey of the development of mutation testing. IEEE Transactions on Software Engineering 37(5), 649–678 (2010)
9. Newman, S.: Building microservices: designing fine-grained systems. O'Reilly Media, Inc. (2015)

The Chances of including Extrinsic Factors in Business Process Management

Christian Sturm

University of Bayreuth
Christian.Sturm@uni-bayreuth.de

Abstract. Business Process Management *(BPM)* focuses on organizing and improving business processes, whereby a process is a set of temporally ordered activities to achieve a defined goal. Additional perspectives besides the activity are considered to better adapt the process environment. However, the process environment is currently restricted to these perspectives. In the era of digitalization, the amount of produced and stored data expanded massively, but the information is not used in *BPM* yet. Hence, it must be investigated how these new information or *extrinsic factors* influence process execution and can be utilized with respect to *BPM* in general.

Keywords: Process Modelling, Process Discovery, Business Process Management, Context Aware Information Systems

1 Introduction and Motivation

A process model contains the execution order of activities of a process as a minimum requirement. However, most of the agile and flexible processes are described more accurate by including additional information. The flexibility of processes is investigated in [7]. To improve *BPM*, multiperspective process models are aiming at a more fine-grained adaption of the process' environment as this environment tend to be in fact rather varying than static. Currently, the multiperspective process model concerns five perspectives including for instance *who* is executing a task, *auxiliary tools* for completing a task or certain *data attributes* which are produced or consumed by activities [5]. The question that arose in this case asks, if the current process design is sufficient or if the consideration of even more information bear chances to lift *BPM* to a new level.

The research area of *BPM* is several years old and certainly the world has underwent a huge change since then. The volume of produced data is exploding reasoning from recent developments in electronic data processing (scalable computation and storage, *IaaS*). Using this technical infrastructure as catalyst, digitalization marks the start of a new era in industry and affects our personal and professional lives. The internetworking of physical devices in production (*IoT*) in conjunction with the acquisition and storage of a vast amount of data opens up new possibilities. The analysis of this *Big Data* has entered many areas

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

in economics and science yet, but is not excessively investigated in *BPM* since now. [6] provides an overview of potential synergy effects concerning *IoT* and *BPM*. The data affecting the process environment is more specified in Sect. 2 and suggestions for a target-oriented analyses w.r.t. *BPM* are proposed.

2 Extrinsic Factors and enclosed Chances and Challenges

To convey an initial feeling for whereof extrinsic factors can be generated from, a few examples of data sources are given in this section with a first proposal of a possible categorization.

Process Related Data is produced near the task execution itself regarding the physical proximity as well as from a logical point of view. For instance, IT-systems can capture information like the specification of devices in use for the task execution (performance indicators, display sizes, operating systems, etc.) or the bandwidth capacity when mobile devices are used within a network.

Organization Internal Data comprises all information belonging to the superior entity of process execution, e.g. a company or departments of government institutes. Examples are the prevailing temperature within a production hall, which affects the manufacturing process (quality of products, cycle times, etc.), conditions of stocks in internal warehouses or shift schedules coming from ERP-systems.

Organization External Data are not further restricted. OpenData-Initiatives follow the trend to provide public access to certain data. Several governments published demographic data, which can be included in process management as well as traffic data or tendencies on (finance) markets. In general, every data source not related to the superior entity of process execution can be used.

The following paragraph points out the dissociation of extrinsic factors and common data attributes. Some characteristics of extrinsic factors are shared with the commonly known data attribute in the multiperspective process model. Despite this certain resemblance, it is totally worth and necessary to distinguish data attributes from extrinsic factors for mainly two reasons.

Semantic Point of View. The core peculiarity of data attributes is the strong correlation to the process instance and the executed activity respectively. For instance, assume a fine management system for road traffic violations. Based on the level of fee to pay, the admonished person may prefer different payment options. The data attribute (fee) is thereby decisive for further execution and further mandatory for a valid process instance. In contrast, extrinsic factors have a rather global point of view on the process and are present even without a certain execution of the process. However, the factors may still influence the process. Moreover, the data attribute constitutes often one single value per one executed activity.

Technical Point of View. Most solutions reach their limits w.r.t. an efficient discovery of a model in the context of Process Mining, especially when considering a second perspective like the organizational perspective or the data attribute. Taking into account much more information will probably cause performance drops so considerations concerning this matter are required.

The following gives an overview of open questions and challenges regarding the implementation of extrinsic factors in *BPM*.

Selection of Extrinsic Factors. One core challenge is the selection of extrinsic factors, which actually do have an impact on the process. As stated, extrinsic factors comprise a variety of possible influence factors, whereas the characteristic of a process is usually restricted to a single domain. The variety forbids a brute-force approach to execute correlation analysis of any factor with the process model regarding the performance – at least with conventional computation methods.

Skilful combinations of multivariate statistical approaches with *Visual Analytics* techniques (see below) must be employed.

Semantic Challenges. What questions can we answer with the help of extrinsic factors? This problem touches almost every application case in the world of data science, and so is necessary to ask in the field of *BPM*. In [2] and [1] the process environment is used to optimize the control flow of processes. As stated, extrinsic factors do not correspond to single activities. Thus, they may also affect different perspectives of activities, like resources or the lifecycle of an execution (throughput or idle times). The extrinsic factors may also prevent processes from its execution while completely different processes are running. It has to be investigated, how current *BPM* supports this variety of information gain or how design or discovery purposes must be adapted.

Technical Challenges. Especially in context of declarative process mining, including extrinsic factors into the calculations poses a challenge regarding the performance. Even without concerning extrinsic factors, most of current solutions cannot finish in reasonable time. Hence, the computing time must be taken into account. A first approach to tackle this issue using *Big Data* technology was presented in [8].

Data Quality. Most of the time when using Data Science technology on a vast volume of data, the analysis is accompanied with trust issues. Algorithms are used as a black box and the computation model often bears a lack of transparency. The input information is badly subjected to quality services, although input information is untrustworthy and wrong or even too less input data causes wrong output information. Furthermore, the quality of the output data can suffer from bias, for instance when using a limited sample set from the whole recorded data in statistical analysis like in [4].

Visual Analytics. Visual Analytics is an interdisciplinary field providing interesting approaches helping analysts to investigate big-sized and especially multivariate data. These ideas can help to support conveying discovered

correlations and dependencies or can even applied in discovery procedures. A first approach is presented in [3].

Process Life Cycle. The vast amount and variety of data is not manageable in the *design phase*. However, in a re-design phase, when extrinsic factors were determined and the influence was disclosed within a preceded process mining step, process models may incorporate extrinsic factors. Therefore, it has to be investigated, if current *modelling* approaches must be extended to be able to denote the additional information. The (technical) difficulties of *process mining* were stated above and in an *optimization* step, extrinsic factors and process information can be included both in machine learning algorithms to automatically submit proposals for improvements to process scientists.

3 Conclusion

Business Process Management focuses on optimizing business processes. However, the high availability of data through digitalization or *IoT* is exploited only marginally since now. Current research considers process environment but shows drawbacks w.r.t. performance, quality of output data or the scope of the analysis. Thus, it has to be investigated how to boost the performance and which additional questions can be answered by *BPM* with extrinsic factors.

References

1. Carvalho, J.d.E.S., et al.: Learning context to adapt business processes. In: CSCWD (2013)
2. Carvalho, J.d.E.S., et al.: A method to infer the need to update situations in business process adaptation. Comput. Ind. 71(C), 128–143 (Aug 2015)
3. Dixit, P.M., et al.: Enabling interactive process analysis with process mining and visual analytics. In: BIOSTEC. pp. 573–584 (2017)
4. Hompes, B.F.A., et al.: A generic framework for context-aware process performance analysis. In: OTM (2016)
5. Jablonski, S., Bussler, C.: Workflow management – modeling concepts, architecture and implementation. International Thomson (1996)
6. Janiesch, C., et al.: The internet-of-things meets business process management: Mutual benefits and challenges. CoRR abs/1709.03628 (2017)
7. Schonenberg, H., et al.: Process flexibility: A survey of contemporary approaches. In: Advances in Enterprise Engineering I (2008)
8. Sturm, C., et al.: Distributed multi-perspective declare discovery. In: BPM Demo Track and BPM Dissertation Award (2017)

Schritte zu einer zertifizierten Informationsflussanalyse von Geschäftsprozessen

Thomas S. Heinze

Institut für Informatik
Friedrich-Schiller-Universität Jena
t.heinze@uni-jena.de

Zusammenfassung. In diesem Beitrag wird eine einfache Methode zur Analyse des Informationsflusses in Geschäftsprozessen auf Grundlage von Vertraulichkeitsstufen vorgestellt. Die Korrektheit der Analyse wird anhand einer maschinenprüfbaren Formalisierung in Coq nachgewiesen.

1 Einführung und Motivation

Die statische Datenflussanalyse ist ein geeignetes Werkzeug zur Unterstützung der Geschäftsprozessmodellierung und -analyse. Wie bereits in [8] ausgeführt, bietet sie nicht nur ein allgemein einsetzbares Rahmenwerk zur Prozessanalyse, sondern beruht gleichzeitig auch auf der wohldefinierten Theorie der *abstrakten Interpretation* [6], die sich leicht zum formalen Korrektheitsnachweis heranziehen lässt. In Verbindung mit modernen maschinengestützten Beweisassistenten gestattet die Formalisierung von Datenflussanalysen auf Grundlage der abstrakten Interpretation schließlich die Definition sogenannter *zertifizierter Analysen*. Grundlage einer solchen zertifizierten Analyse bildet dabei ein maschinenprüfbarer Beweis der Analysekorrektheit, aus dem sich die Analyseimplementierung extrahieren lässt. Der Vorteil liegt dann auf der Hand, da ein Anwender nun nicht mehr auf die Korrektheit der Implementierung vertrauen muss, sondern stattdessen den Korrektheitsbeweis automatisiert nachvollziehen kann. In einem Auditszenario ließe sich derart die Compliance-Prüfung von Geschäftsprozessen [2] durch eine orthogonale Prüfung zur Vertrauenswürdigkeit des Audits selbst ergänzen.

Ein Beispiel für ein solches Szenario ist die Analyse von Informationsflüssen, oder genauer die Untersuchung des Flusses von Informationen in einem Prozessmodell, mit der sich etwa Compliance-Regeln zur vertraulichen Verarbeitung von sensitiven Informationen, beispielsweise Gesundheitsdaten, überprüfen lassen. In diesem Beitrag wird eine erste einfache statische Datenflussanalyse zur Analyse des Informationsflusses in Geschäftsprozessen auf Grundlage von Vertraulichkeitsstufen vorgestellt. Die Analyse beruht auf dem Verfahren der Andersen-Analyse [16] und wurde mit Hilfe des Beweisassistenten *Coq*[1] formal definiert, so dass der Nachweis der Analysekorrektheit als maschinenprüfbarer Beweis zur Verfügung steht. Der Beitrag kann somit als ein Schritt zu einer *zertifizierten Informationsflussanalyse* von Geschäftsprozessen verstanden werden.

[1] https://coq.inria.fr, letzter Zugriff am 8. März 2018

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

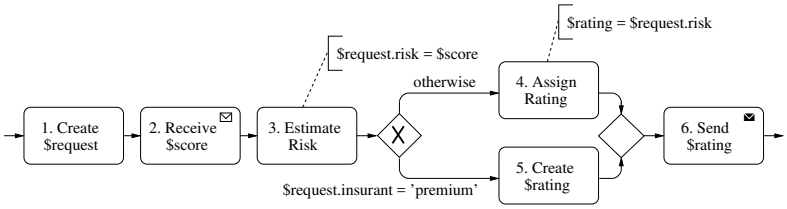

Abb. 1. Beispielprozess in BPMN-Notation

Im sich anschließenden Abschnitt 2 wird das Analyseszenario genauer abgegrenzt und die Analyse vorgestellt. Die formale Entwicklung und der Korrektheitsnachweis zur Analyse auf Grundlage der Theorie der abstrakten Interpretation erfolgt in Abschnitt 3. In Abschnitt 4 wird ein kurzer Überblick zu verwandten Arbeiten gegeben, bevor der Beitrag mit einer Zusammenfassung und einem Ausblick auf weiterführende Arbeiten in Abschnitt 5 endet.

2 Analyse des Informationsflusses

Eine *Informationsflussanalyse* [7] untersucht einen Prozess hinsichtlich des Flusses von Informationen, wobei verschiedene Vertraulichkeitsstufen berücksichtigt werden können. Im einfachsten Fall werden die zwei Stufen H (vertraulich) und L (öffentlich) betrachtet. Zur Sicherstellung der Vertraulichkeit sollte dann kein Informationsfluss von einer mit H ausgezeichneten Quelle an eine mit L ausgezeichnete Senke existieren. In Abbildung 1 ist ein Beispielprozess dargestellt, in dem die Risikobewertung für eine Versicherung bestimmt und versendet wird. Dazu wird zunächst ein Formular in $\$request$ erzeugt und der vertrauliche Score-Wert $\$score$ abgefragt. In Abhängigkeit vom Versichertenstatus wird entweder ein Standardwert oder der Score-Wert als Risikobewertung $\$rating$ verwendet und weitergeleitet. Wird in diesem Beispiel die eingehende Nachricht $\$score$ mit der Vertraulichkeitsstufe H und die ausgehende Nachricht $\$rating$ mit der Stufe L ausgezeichnet, liegt für den oberen Pfad ein Informationsfluss von H nach L und somit eine Verletzung der Informationsvertraulichkeit vor.

Hier soll eine statische Datenflussanalyse zur Untersuchung des Informationsflusses in Prozessmodellen definiert werden. Gegenstand der Analyse sind explizite Datenflüsse, wie im obigen Beispiel, jedoch keine impliziten Informationsflüsse oder Seitenkanäle (vergleiche den impliziten Fluss von $\$request.insurant$ nach $\$rating$ über die Kontrollabhängigkeit der Variablendefinition). Unter diesen Anforderungen lässt sich die Analyse als *Taint-Analyse* [14] auffassen und auf das Verfahren der Andersen-Analyse [16] zurückführen. Durch dieses Verfahren werden Variablen, und deren Komponenten im Fall zusammengesetzter Variablentypen, Mengen mit abstrakten Objekten zugewiesen. Weiterhin werden Regeln zwischen den Mengen in Form von Element-/Teilmengenbeziehungen definiert. Die Lösung des dadurch charakterisierten Regelsystems ergibt dann einen Fixpunkt als Abschätzung zum Datenfluss im analysisertem Prozess.

$$
\begin{aligned}
i\colon \mathit{Receive}^L\ x/\ i\colon \mathit{Create}^L\ x : &\quad o_i^L \in pt(x)\\
i\colon \mathit{Receive}^H\ x/\ i\colon \mathit{Create}^H\ x : &\quad o_i^H \in pt(x)\\
x = y : &\quad pt(y) \subseteq pt(x)\\
x = y.f : &\quad \frac{o_i^L \in pt(y)}{pt(o_i.f) \subseteq pt(x)} \qquad \frac{o_i^H \in pt(y)}{pt(o_i.f) \subseteq pt(x)}\\
x.f = y : &\quad \frac{o_i^L \in pt(x)}{pt(y) \subseteq pt(o_i.f)} \qquad \frac{o_i^H \in pt(x)}{pt(y) \subseteq pt(o_i.f)}
\end{aligned}
$$

Abb. 2. Regelsystem zur statischen Informationsflussanalyse

In Abbildung 2 sind die Regeln der Analyse für einen kleinen Sprachausschnitt dargestellt. Auf der linken Seite stehen die Prozessaktivitäten und auf der rechten Seite die zugehörigen Regelschemata. Zusätzlich zu einer herkömmlichen Andersen-Analyse wird hier zwischen öffentlichen und vertraulichen Informationsquellen unterschieden, so dass die Aktivitäten *Receive* und *Create* eine entsprechende Auszeichnung mit L oder H aufweisen müssen. Komplexere Sprachkonstrukte, etwa die Zuweisung $x.f.g = y.h$, lassen sich durch die Einführung zusätzlicher Variablen ebenfalls abbilden. Ferner wird von atomaren Nachrichten ausgegangen.

Durch die Regeln wird für jede Prozessaktivität beschrieben, wie deren Ausführung die den Variablen und Komponenten zugeordneten Mengen pt mit abstrakten Objekten beeinflusst. Als Fixpunktlösung ergibt sich für die Regeln dann eine Überabschätzung des Datenflusses, da Ausführungsreihenfolge und -kontext von Aktivitäten unberücksichtigt bleiben (*fluss-/kontextinsensitiv* [16]). Zudem wird pro *Receive*- und *Create*-Aktivität i nur jeweils ein abstraktes Objekt o_i unterschieden, unabhängig davon wie oft die Aktivität ausgeführt wird. Wird der Beispielprozess in Abbildung 1 betrachtet, und von einer vertraulichen Quelle $\$score$ sowie einer öffentlichen Quelle $\$request$ ausgegangen, ergibt sich unter Anwendung der Regeln die folgende Ableitung und somit eine Vertraulichkeitsverletzung, unter Annahme der Vertraulichkeitsstufe L für die ausgehende Nachricht $\$rating$ (Regelanwendungen sind mit der zugehörigen Aktivitäts-Id angegeben):

$$
\dfrac{\dfrac{\dfrac{}{o_1^L \in pt(\$request)}\,{\scriptstyle 1}}{pt(o_1.risk) \subseteq pt(\$rating)}\,{\scriptstyle 4} \qquad \dfrac{\dfrac{\dfrac{}{o_1^L \in pt(\$request)}\,{\scriptstyle 1}}{pt(\$score) \subseteq pt(o_1.risk)}\,{\scriptstyle 3} \qquad \dfrac{}{o_2^H \in pt(\$score)}\,{\scriptstyle 2}}{o_2^H \in pt(o_1.risk)}}{o_2^H \in pt(\$rating)}
$$

3 Zertifizierte Informationsflussanalyse

Im Folgenden soll die im vorangegangenen Abschnitt vorgestellte Analyse formal definiert und deren Korrektheit bewiesen werden. Die Formalisierung erfolgte mit Hilfe des Beweisassistenten *Coq*, so dass ein maschinenprüfbarer Beweis zur

$$\mathit{Receive}^L\ x/\ \mathit{Create}^L\ x,\ (\mathit{vars}, \mathit{heap}, \mathit{high}, l) \to (\mathit{vars}[x \leftarrow l+1], \mathit{heap}, \mathit{high}, l+1)$$

$$\mathit{Receive}^H\ x/\ \mathit{Create}^H\ x,\ (\mathit{vars}, \mathit{heap}, \mathit{high}, l) \to (\mathit{vars}[x \leftarrow l+1], \mathit{heap}, \mathit{high} \cup \{l+1\}, l+1)$$

$$x = y,\ (\mathit{vars}, \mathit{heap}, \mathit{high}, l) \to (\mathit{vars}[x \leftarrow \mathit{vars}(y)], \mathit{heap}, \mathit{high}, l)$$

$$\frac{\mathit{heap}(\mathit{vars}(y)) \neq 0}{x = y.f,\ (\mathit{vars}, \mathit{heap}, \mathit{high}, l) \to (\mathit{vars}[x \leftarrow \mathit{heap}(\mathit{vars}(y))], \mathit{heap}, \mathit{high}, l)}$$

$$\frac{\mathit{heap}(\mathit{vars}(y)) = 0}{x = y.f,\ (\mathit{vars}, \mathit{heap}, \mathit{high}, l) \to (\mathit{vars}, \mathit{heap}, \mathit{high}, l)}$$

$$\frac{\mathit{vars}(x) \neq 0}{x.f = y,\ (\mathit{vars}, \mathit{heap}, \mathit{high}, l) \to (\mathit{vars}, \mathit{heap}[\mathit{vars}(x) \leftarrow \mathit{vars}(y)], \mathit{high}, l)}$$

$$\frac{\mathit{vars}(x) = 0}{x.f = y,\ (\mathit{vars}, \mathit{heap}, \mathit{high}, l) \to (\mathit{vars}, \mathit{heap}, \mathit{high}, l)}$$

Abb. 3. Konkrete Semantik (Prädikat `[exec]` in Coq-Formalisierung)

Analysekorrektheit[2] zur Verfügung steht und die Implementierung der Analyse aus dem Beweis extrahiert werden kann. Der Coq-Formalismus beruht dabei wesentlich auf bestehenden Coq-Quellen[3] zur Andersen-Analyse.

Für die formale Entwicklung einer statischen Analyse kann auf die *abstrakten Interpretation* zurückgegriffen werden [6]. In der abstrakten Interpretation wird ein Prozess nicht auf konkreten sondern auf abstrakten Werten ausgeführt. Anstatt mit *int*-Werten wird etwa mit den Werten *odd* und *even* gerechnet, so dass sich die Addition aus den Regeln $\mathit{even} + \mathit{even} = \mathit{even}$, $\mathit{even} + \mathit{odd} = \mathit{odd}$, $\mathit{odd} + \mathit{even} = \mathit{odd}$ und $\mathit{odd} + \mathit{odd} = \mathit{even}$ ergibt. Die Ausführung auf konkreten Werten definiert die *konkrete Semantik* und die Ausführung auf abstrakten Werten die *abstrakte Semantik*. Letztere wird zur Definition der Analyse verwendet, für die ausgehend von einem Startzustand durch kontinuierliche Anwendung der abstrakten Semantik auf einen Prozess ein Fixpunkt errechnet wird. Zum Nachweis der Korrektheit muss gezeigt werden, dass der Fixpunkt die sich für jeden Zustand aus der konkreten Semantik ergebenden konkreten Werte abschätzt.

Zur Formalisierung der Informationsflussanalyse wird die in Abbildung 3 dargestellte *konkrete Semantik* genutzt. Darin werden Variablen, Komponenten und Objekte als natürliche Zahlen kodiert. Ein Zustand s enstpricht einem Tupel $(\mathit{vars}, \mathit{heap}, \mathit{high}, l)$, wobei $\mathit{vars}, \mathit{heap}\colon \mathbb{N} \to \mathbb{N}$ Variablen und Komponenten die Objekte zuordnet, auf die sie im Zustand s verweisen, $\mathit{high} \subseteq \mathbb{N}$ alle vertraulichen Objekte aufzählt und $l \in \mathbb{N}$ das zuletzt erzeugte Objekt bezeichnet. Die konkrete Semantik definiert damit eine Relation $i, s \to s'$ zwischen einem Zustand s, einer auf diesem ausgeführten Prozessaktivität i und dem sich daraus ergebenden Folgezustand s'. Durch die Relation wird beispielsweise für eine Aktivität $\mathit{Create}^H\ x$ im Folgezustand ein neues Objekt $l+1$ erzeugt, der Menge vertraulicher Objekt hinzugefügt und der Verweis der Variablen x auf das Objekt $l+1$ gesetzt.

[2] `https://gitlab.com/t.heinze/zeus2018.git`

[3] `http://adam.chlipala.net/itp/coq/src`, letzter Zugriff am 8. März 2018

$$\begin{aligned}
&\mathit{Receive}^L\ x\#k/\ \mathit{Create}^L\ x\#k,\ (\mathit{avars}, \mathit{aheap}, \mathit{ahigh})\\
&\qquad \to (\mathit{avars}[x \leftarrow \mathit{avars}(x) \cup \{k\}], \mathit{aheap}, \mathit{ahigh})\\
&\mathit{Receive}^H\ x\#k/\ \mathit{Create}^H\ x\#k\ (\mathit{avars}, \mathit{aheap}, \mathit{ahigh})\\
&\qquad \to (\mathit{avars}[x \leftarrow \mathit{avars}(x) \cup \{k\}], \mathit{aheap}, \mathit{ahigh} \cup \{k\})\\
&x = y,\ (\mathit{avars}, \mathit{aheap}, \mathit{ahigh}) \to (\mathit{avars}[x \leftarrow \mathit{avars}(x) \cup \mathit{avars}(y)], \mathit{aheap}, \mathit{ahigh})\\
&x = y.f,\ (\mathit{avars}, \mathit{aheap}, \mathit{ahigh})\\
&\qquad \to (\mathit{avars}[x \leftarrow \mathit{avars}(x) \cup \bigcup_{k \in \mathit{avars}(y)} \mathit{aheap}(k)], \mathit{aheap}, \mathit{ahigh})\\
&x.f = y,\ (\mathit{avars}, \mathit{aheap}, \mathit{ahigh})\\
&\qquad \to (\mathit{avars}, \mathit{aheap}[k \leftarrow \mathit{aheap}(k) \cup \mathit{avars}(y) \mid k \in \mathit{avars}(x)], \mathit{ahigh})
\end{aligned}$$

Abb. 4. Abstrakte Semantik (Prädikat `[abstract_exec]` in Coq-Formalisierung)

In der *abstrakten Semantik* in Abbildung 4 werden abstrakte Zustände betrachtet, die Tupel $(\mathit{avars}, \mathit{aheap}, \mathit{ahigh})$ sind, wobei $\mathit{avars}, \mathit{aheap}\colon \mathbb{N} \to \mathcal{P}(\mathbb{N})$ nun die möglichen Verweisziele von Variablen und Komponenten in Form von Mengen abschätzen (vergleiche auch Abbildung 2), *ahigh* fasst alle vertraulichen Objekte zusammen. Im Gegensatz zur konkreten Semantik wird nicht mehr für jede Ausführung einer *Receive*- oder *Create*-Aktivität ein neues Objekt erzeugt, sondern für dieselbe Aktivität nur ein Objekt verwendet. Die Aktivitäten werden dazu durchnummeriert, so dass sich das Objekt aus dem Index k ergibt.

Für den Nachweis der Korrektheit werden die konkrete und die abstrakte Semantik in Beziehung gesetzt, dies mittels sogenannter Objektpfade:

$$\frac{s = (\mathit{vars}, \mathit{heap}, \mathit{high}, l)}{s \vdash v :: \mathit{vars}[v]} \qquad \frac{s = (\mathit{vars}, \mathit{heap}, \mathit{high}, l) \quad s \vdash p :: l' \quad l' \neq 0}{s \vdash p :: l' :: \mathit{heap}[l']}$$

$$\frac{a = (\mathit{avars}, \mathit{aheap}, \mathit{ahigh}) \quad k \in \mathit{avars}[v]}{a \vdash v :: k}$$

$$\frac{a = (\mathit{avars}, \mathit{aheap}, \mathit{ahigh}) \quad a \vdash p :: k \quad k' \in \mathit{aheap}[k]}{a \vdash p :: k :: k'}$$

Ein Objektpfad $v :: p :: n$ ist ein Prädikat, das ausgehend von einer Variablen v, unter einen gegebenen konkreten Zustand s beziehungsweise abstrakten Zustand a, eine eventuell leere Sequenz p von Kompontenenzugriffen auf ein Objekt n beschreibt. Vereinfacht gesprochen bezeichnet ein Objektpfad die Möglichkeit unter Zustand s beziehungsweise a über die Variable v auf das Objekt n zuzugreifen. Wie in Abbildung 5 definiert, ist ein abstrakter Zustand a dann eine Abschätzung für einen konkreten Zustand s, falls zusätzlich zu gewissen Nebenbedingungen folgende zwei Bedingungen erfüllt sind: (1) Für jedes Objekt l' auf das im Zustand s über Objektpfade $v_1 :: p :: l'$ und $v_2 :: q :: l'$ zugegriffen werden kann, existiert im Zustand a ein entsprechendes Objekt k auf das über Objektpfade $v_1 :: p' : k$ und $v_2 :: q' :: k$ zugegriffen werden kann. (2) Für jedes vertrauliche Objekt l' auf das über einen Objektpfad $v :: p :: l'$ im Zustand s zugegriffen werden kann, existiert im Zustand a ein entsprechendes vertrauliches Objekt k auf das über einen Objektpfad $v :: q :: k$ zugegriffen werden kann. Anhand dieser Definitionen

$a = (avars, aheap, ahigh)$ approximates $s = (vars, heap, high, l)$
$\Leftrightarrow_{df}$ $heap(0) = 0 \wedge \forall l' > l \colon heap(l') = 0 \wedge \forall l' \colon s \vdash p :: l' \Rightarrow l' \leq l \wedge \forall l' \in high \colon l' \leq l$
$\wedge \forall v_1, v_2, l' \neq 0 \colon s \vdash v_1 :: p :: l' \wedge v_2 :: q :: l' \Rightarrow \exists k \colon a \vdash v_1 :: p' :: k \wedge a \vdash v_2 :: q' :: k$
$\wedge \forall v, l' \neq 0 \colon s \vdash v :: p :: l \wedge l \in high \Rightarrow \exists k \colon a \vdash v :: q :: k \wedge k \in ahigh$

Konservativität der abstrakten Semantik:

$\forall a', s', s \colon s' \rightsquigarrow s \wedge a'$ approximates $s' \Rightarrow \exists a \colon a' \rightsquigarrow a \wedge a$ approximates s □

v not $H \Leftrightarrow_{df} \forall s = (vars, heap, high, l) \colon init \rightsquigarrow s \Rightarrow vars(v) \notin high$

Korrektheit der Informationsflussanalyse:

$\forall v \colon (\forall a = (avars, aheap, ahigh) \colon ainit \rightsquigarrow a \Rightarrow \forall k \in avars(v) \colon k \notin ahigh) \Rightarrow v$ not H □

Abb. 5. Korrektheit der Analyse (Theorem `[analysis_sound]` in Coq-Formalisierung)

lässt sich zeigen, dass die abstrakte Semantik eine konservative Abschätzung für die konkrete Semantik ist, das heißt für alle erreichbaren konkreten Zustände s existiert ein erreichbarer abstrakter Zustand a, der eine Abschätzung für s ist.

Als Folgerung aus dieser Beziehung zwischen konkreter und abstrakter Semantik kann auf die Korrektheit der Informationsflussanalyse geschlossen werden. So lässt sich zeigen, dass für alle Variablen v gilt, falls v unter allen erreichbaren abstrakten Zuständen a auf kein vertrauliches Objekt verweist, so verweist v auch unter allen erreichbaren konkreten Zuständen s auf kein vertrauliches Objekt. Unter Anwendung der Analyse kann somit ein Fluss von einer vertraulichen Quelle an eine öffentliche Senke, in Form einer Variablen v, sicher ausgeschlossen werden. Neben der Korrektheit der Analyse kann auch deren Terminierung gezeigt werden. Aus Platzgründen wird auf eine entsprechende Diskussion verzichtet und stattdessen auf die Coq-Formalisierung (Definition `[fixed_point]`) verwiesen.

4 Verwandte Arbeiten

In [8] wurde der Ansatz zur zertifizierten Analyse von Geschäftsprozessen zunächst allgemein motiviert, in diesem Beitrag konnte die Machbarkeit des Ansatzes nun konkret am Beispiel der Analyse des Informationsflusses von Geschäftsprozessen gezeigt werden. Die Prüfung des Informationsflusses ist dabei ein Standardproblem der statischen Analyse [7]. Es gibt zahlreiche Varianten, die neben expliziten Datenflüssen auch Seitenkanäle, zum Beispiel das Zeitverhalten oder den Energieverbrauch, einbeziehen. Informationsflussanalysen werden neben dem hier betrachteten Aufdecken von Informationslecks insbesondere zur Identifizierung von Sicherheitslücken eingesetzt, oft in Form einer *Taint-Analyse* [14]. Eine geläufige Technik zur Taint-Analyse ist das Andersen-Verfahren [16]. Für dieses wird zwischen inter- und intraprozeduralen, fluss- und kontextsensitiven sowie -insensitiven Varianten unterschieden. Das hier beschriebene Verfahren setzt eine sehr einfache intraprozedurale und fluss-/kontextinsensitive Analyse um.

Der Einsatz von Beweisassistenten zur Verifikation von statischen Analysen erfährt eine stetig wachsende Verbreitung. Insbesondere *Coq* spielt eine herausragende Rolle, wie nicht zuletzt an der erfolgreichen Verifikation eines realistischen

optimierenden Compilers im *CompCert*-Projekt [13] deutlich wird. Im Allgemeinen wird zum Nachweis der Analysekorrektheit analog dem Vorgehen in diesem Beitrag auf die Theorie der abstrakten Interpretation zurückgegriffen [4,5]. Neben den auch dieser Arbeit zugrundeliegenden Coq-Quellen zum Andersen-Verfahren von Adam Chlipala existieren weitere entsprechende Formalisierungen [11,15].

Eine Reihe von typischerweise petrinetzbasierten Techniken zur Analyse des Informationsflusses wird auch für Geschäftsprozesse beschrieben, eine Übersicht kann etwa [1] entnommen werden. Die hier vorgestellte Analyse bezieht sich dabei auf den expliziten Datenfluss und Vertraulichtkeitsstufen (*Mandatory Access Control*). Vergleichbare Arbeiten für Geschäftsprozesse beruhen meist auf höheren Petrinetzen und Methoden des Model-Checking [3,12]. Damit ergeben sich aber zwei Problemstellungen, die für den in dieser Arbeit beschriebenen Ansatz nicht auftreten. So stellen sich für das Model-Checking auf höheren Petrinetzen aufgrund potentieller Zustandsraumexplosion grundlegende Skalierungsprobleme. Durch eine geschickte Wahl der Abstraktion in den Petrinetzmodellen lassen sich diese zwar prinzipiell umgehen, jedoch bedingt dies dann eine aufwendige und nicht triviale manuelle Modellierung der zu analysierenden Geschäftsprozesse. Ferner ist dem Autor kein Ansatz zur zertifizierten Informationsflussanalyse, das heißt maschinell verifizierten Analyse, für Geschäftsprozesse bekannt.

5 Zusammenfassung und Ausblick

In diesem Beitrag wird eine einfache statische Analyse zur Untersuchung des Informationsflusses in Geschäftsprozessen vorgestellt. Mit Hilfe der Analyse lassen sich Datenflüsse von sensitiven Datenquellen an öffentliche Senken für einen Prozess sicher ausschließen. Die Analyse beruht auf dem Verfahren der Andersen-Analyse und ist mittels der Theorie der abstrakten Interpretation formalisiert, so dass sich Korrektheit und Terminierung im Beweisassistent *Coq* maschinenprüfbar beweisen lassen. Da zudem die Möglichkeit besteht, die Analyseimplementierung aus dem Beweis zu extrahieren, handelt es sich um einen ersten Schritt zu einer *zertifizierten Informationsflussanalyse* für Geschäftsprozesse.

In weiterführenden Arbeiten sollen Fragen zur Präzision und Skalierbarkeit der Analyse untersucht werden. So kann die Präzision durch Definition einer flusssensitiven Analyse erhöht werden, etwa unter Verwendung *erweiterter Workflow-Graphen* [9,10]. In diesem Fall ist aber eine Formalisierung der Transformation in erweiterte Workflow-Graphen in *Coq* notwendig.

Danksagung. Ich danke Adam Chlipala für die Bereitstellung der Coq-Quellen zur Andersen-Analyse.

Literatur

1. Accorsi, Rafael ; Lehmann, Andreas ; Lohmann, Niels: Information leak detection in business process models: Theory, application, and tool support. In: *Information Systems* 47 (2015), S. 244–257

2. Accorsi, Rafael ; Lowis, Lutz ; Sato, Yoshinori: Automated Certification for Compliant Cloud-based Business Processes. In: *Business & Information Systems Engineering* 3 (2011), Nr. 3, S. 145–154
3. Barkaoui, Kamel ; Ayed, Rahma B. ; Boucheneb, Hanifa ; Hicheur, Awatef: Verification of Workflow Processes Under Multilevel Security Considerations. In: *Proceedings, Third International Conference on Risks and Security of Internet and Systems, CRiSIS 2008, Tozeur, Tunisia, October 28-30, 2008*, IEEE, 2008, S. 77–84
4. Bertot, Yves: Structural Abstract Interpretation: A Formal Study Using Coq. In: *Language Engineering and Rigorous Software Development, International LerNet ALFA SummerSchool 2008, Piriapolis, Urugay, February 24-March 1, 2008, Revised Tutorial Lectures*. Springer, 2009 (LNCS 5520), S. 153–194
5. Besson, Frédéric ; Cachera, David ; Jensen, Thomas ; Pichardie, David: Certified Static Analysis by Abstract Interpretation. In: *Foundations of Security Analysis and Design V, FOSAD 2007/2008/2009 Tutorial Lectures*. Springer, 2009 (LNCS 5705), S. 223–257
6. Cousot, Patrick ; Cousot, Radhia: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs. In: *Proceedings of the 4th ACM Symposium on Principles of Programming Languages*, ACM, 1977, S. 238–252
7. Denning, Dorothy E.: A Lattice Model of Secure Information Flow. In: *Communications of the ACM* 19 (1976), Nr. 5, S. 236–243
8. Heinze, Thomas S.: Towards Certified Data Flow Analysis of Business Processes. In: *9th ZEUS Workshop, ZEUS 2017, Lugano, Switzerland, 13–14 February 2017, Proceedings*, CEUR-WS.org, 2017 (CEUR Workshop Proceedings 1826), S. 1–3
9. Heinze, Thomas S. ; Amme, Wolfram ; Moser, Simon: A Restructuring Method for WS-BPEL Business Processes Based on Extended Workflow Graphs. In: *Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009, Proceedings*, Springer, 2009 (LNCS 5701), S. 211–228
10. Heinze, Thomas S. ; Amme, Wolfram ; Moser, Simon: Static analysis and process model transformation for an advanced business process to Petri net mapping. In: *Software: Practice and Experience* 48 (2018), Nr. 1, S. 161–195
11. Jourdan, Jacques-Henri ; Laporte, Vincent ; Blazy, Sandrine ; Leroy, Xavier ; Pichardie, David: A Formally-Verified C Static Analyzer. In: *ACM SIGPLAN Notices* 50 (2015), Nr. 1, S. 247–259
12. Juszczyszyn, Krzysztof: Verifying Enterprise's Mandatory Access Control Policies with Coloured Petri Nets. In: *Proceedings, Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE 2003, June 9-11, 2003, Johannes Kepler University of Linz, Austria*, IEEE, 2003, S. 184–189
13. Leroy, Xavier: Formal Verification of a Realistic Compiler. In: *Communications of the ACM* 52 (2009), Nr. 7, S. 107–115
14. Livshits, V. B. ; Lam, Monica S.: Finding Security Vulnerabilities in Java Applications with Static Analysis. In: *Proceedings of the 14th USENIX Security Symposium*, USENIX Association, 2005, S. 271–286
15. Robert, Valentin ; Leroy, Xavier: A Formally-Verified Alias Analysis. In: *Certified Programs and Proofs, Second International Conference, CPP 2012, Kyoto, Japan, December 13-15, 2012, Proceedings*, Springer, 2009 (LNCS 7679), S. 11–26
16. Sridharan, Manu ; Chandra, Satish ; Dolby, Julian ; Fink, Stephen J. ; Yahav, Eran: Alias Analysis for Object-Oriented Programs. In: *Aliasing in Object-Oriented Programming*. Springer, 2013 (LNCS 7850), S. 196–232

RBPSim: A Resource-aware Extension of BPSim Using Workflow Resource Patterns

Nehal Afifi, Ahmed Awad and Hisham M. Abdelsalam

Faculty of Computers and Information, Cairo University, Giza, Egypt
n.afifi@grad.fci-cu.edu.eg,{a.gaafar,h.abdelsalam}@fci-cu.edu.eg

Abstract. One of the main limitations affecting business process simulation approaches is the incorrect modeling of human resources. The BPSim standard is acknowledged as a first step towards streamlining the experience of business process simulation and providing a tool independent exchange format for so-called simulation scenarios. Unfortunately, with respect to the human resource perspective, the standardization effort did not advance behind modeling resources as quantities required for the different process elements. Workflow resource patterns outline resources' distribution and utilization. This paper is taking a first step towards combining BPSim standard with the well-known workflow resource patterns through RBPSim: a resource-aware extension of BPSim standard.

Keywords: BPSim, workflow resource patterns, business process simulation

1 Motivation

Analysis of business process models has for a long time focused on verification, e.g. soundness [1], and validation, e.g. compliance checking [4]. Simulation as an important analysis approach for business processes has received very little attention from researchers [9]. Simulation attempts to predict how the real-world processes will operate through various "What-if" scenarios [3, 5].

For simulating business processes, we need to model at least three perspectives: control flow, data and resources [2]. One of the main issues affecting the current business process simulation approaches is modeling human resource in a naïve manner [2]. Resources are referred to either by quantities within a role or explicitly. To enrich simulation models with appropriate specifications of resource requirements, we need a standard definition of a resource model to be used in any Business Process Simulation(BPS) experiment [7].

The Business Process Simulation (BPSim) standard version 2.0 [16], developed by WfMC, allows business process models specified in either BPMN [10] or XPDL [15] to be augmented with simulation-specific parameters such as task durations, branching probabilities, case arrival rates, etc. The BPSim metamodel is not fully elaborated regarding the resource perspective. Oversimplified resource modeling and omitting workflow resource patters can cause unsuccessful simulations [9, 9, 12]. Resources require richer representation with respect to work

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

preference, speed and realistic allocation plan and working schedule regard to work items. BPSim limitations in modeling resources have been discussed in [6,8]. However, the standard is extensible as it defines a meta-model for its elements and extension points.

In this paper, we take a first step towards extending BPSim to enrich resources specification. We use the well-known workflow resource patterns [12] as the means to express resource's selection and allocation strategies during a simulation scenario. The rest of the paper is organized as the following: section 2 discusses extending BPSim standard with workflow resource patterns, an evaluation example is presented in Section 3. Section 4 concludes the paper with an outlook on future work.

2 Extending BPSim with Workflow Resource Patterns

In this section we discuss the level of support offered by BPSim regarding human resources. We present our extension of BPSim metamodel for more expressive resources representation in simulation scenarios. This extension is based on the well-known workflow resource patterns [12,13].

2.1 Revisiting Workflow Resource Patterns

Workflow resource patterns specify resources representation, selection and utilization within the process model and are divided into seven groups [12,13]. **1. Creation patterns** are concerned with *which resources are eligible?*. That is out of all available resources R, a Creation pattern cp is responsible for finding set $R_{cp} \subseteq R$ which represents the candidate resources where any of them can execute the respective task t. R_{cp} can be either specified by properties that each resource $r \in R_{cp}$ must possess to be able to execute t or it can be specified by explicitly enumerating its members; **2. Push patterns** are concerned with *How to pick one of the eligible resources?*. Push patterns are more on the execution or simulation time assignment of a work item wi to a resource $r \in R_{cp}$ where wi is the instance of task t within a specific process instance. So, the enforcement of a Push pattern should result in at most one specific resource being assigned to the work item whereas enforcement of a Creation pattern results in a set of candidate resources R_{cp}. Note that R_{cp} might be empty in case of none of the available resources possesses sufficient capabilities to perform t; **3. Pull patterns** the difference between Push and Pull patterns is that transitions in Push patterns are stated by the system while in Pull patterns resources may have the ability to initiate transitions, reorder their working queue and select the next work item to be executed; **4. Detour patterns** refer to interruptions to work items either by the system or by resources executing them; **5. Auto-Start patterns** refer to the triggering of work items by specific events either through creation or allocation; **6. Visibility patterns** determining work item visibility for a resource and **7. Multiple Resource patterns** are concerned with tasks that require more than one resource working on it concurrently.

Following a divide-and-conquer approach, in this paper, we are concerned with the first five groups, namely Creation, Push, Pull, Detour and Auto-Start patterns. We argue that building simulation models where those patterns can be employed is a first step towards getting more accurate simulation results. Multiple Resource patterns are currently out of scope and subject for future research. Visibility patterns are purely related to process enactment and building of process execution engines, like implementation of work item lists. Thus, they are not considered for process simulation.

2.2 BPSim Resources Representation Limitations

To accomplish an effective business process simulation experiment, process elements, e.g. tasks, involved in the experiment should include the following [6,14]: **1. Required resources:** to execute a task, one or more resources should be available to handle the task based on task-specific requirements; **2. Execution duration:** resources execution duration for each task is not constant and should follow a probabilistic distribution; **3. Resource share-ability:** resources are not dedicating all their time to one task and may divide time simultaneously between different tasks; **4. Resource availability:** resources may be unavailable to perform tasks which requires a value attached to the resources indicating availability; **5. Context switching overhead**: resources may require time intervals between different tasks execution so another timing interval with a probabilistic distribution should be included to specify time required between tasks execution and **6. Work item selection:** resource's working queue is necessary to organize work items and specify how resource will select the next work item (FIFO, LIFO or based on priority). Among the above-mentioned requirements for a simulation experiment, BPSim covers the following [16]:

1. ElementParameter indicates the reference to a process element and extended with a number of parameters: (a) time, (b) control that defines control flow of BPMN element, (c) cost, (d) priority contains "Interruptible Attribute" specifies if the execution is interruptible and "Priority Attribute" defines the resource allocation order based on element priority, (e) property (f) expressions are added functions such as *getResource* to select a collection of available resources, *getResourceByRole* to select a collection of available resources based on role and *Resource* to select an alternative list of available resources and (g) resource parameter. *2. ResourceParameter* specifies the resource's availability, quantity, selection based on defined role list or a number referring to a specific resourceID. Availability, quantity and role properties are only applicable for an individual resource element leading to the inability of selecting a specific resources based on other criteria. *3. TimeParameter and ControlParameter* are neither related to a resource nor resourcesRole elements although in some cases it might be needed, TimeParameter could be added to resource and resourceRole BPMN elements. *4. CostParameter* could be applied to the resource element (but not the resourceRole element) specifying the cost of resources either by fixed cost attribute based on resource usage or unit cost attribute based on a time unit [6]. *5. PriorityParameter and PropertyParameter* are not applicable for resource and

resourceRole elements. Condition attribute in ControlParameter is just a Boolean and only one condition could be applied to a BPMN element, this parameter should be modified to handle an expression parameters for filtering the resource required for task execution.

Several tools support business process simulation. They are either scientific prototypes, e.g., BIMP [1] and Desmo-J [11] or commercial tools, e.g., Bizagi [2], BonitaSoft [3], Visual Paradigm [4] and Trisotech Modeler [5]. Some of these tools are BPSim compliant. Freitas et al. [6], have assessed these tools with respect to their BPS capabilities. Concerning the resources perspective, all the selected tools support setting the number of available resources to execute work items, only Bizagi and Trisotech support allocation plan of resources, none of the tools supports unavailability of resources and all tools except Visual Paradigm define the resources working schedule and the resources usage cost. All tools define task cost while only Trisotech supports defining task execution priority and ability of interruption while running.

All mentioned tools that go beyond the resources support in BPSim still are not fully covering the resource's perspective. In Section 2.3 we discuss BPSim extended metamodel and provide a systematic way for more expressive resource representation in simulation scenario.

2.3 Realizing Workflow Resources Patterns in BPSim

In this section, we demonstrate our extension of BPSim metamodel showing how to address workflow resource patterns discussed in Section 2.1 based on the BPSim standard. Implementaion of the extended metamodel is left for future work. The extended metamodel is shown in Fig. 1. The newly added classes are highlighted with gray.

BPSim introduces the scenario entity containing all parameters needed to run a simulation. Each scenario represents one *what-if* case. Thus, it is defined for each business process element, e.g. a task, several parameters including, duration, resources, time unit etc. **Resources:** entity is the parent for both human and non-human resources, in this paper we are concerned with **HumanResources** to enumerate resources participating in a scenario and describe them with attributes. Those resources might be referred to later directly or indirectly by a resourceParameter. **ResourceQueue:** is used to handle resource's work items waiting to be executed. Work items in the resource's queue could be sorted based on preference of specific work items, FIFO, priority, LIFO. **Role:** A human resource is a member of role which generalizes over **OrganizationalGroup** and **Position** that has (**Privileges**). A human resource may have execution history of work items **History** and a **ShiftCalendar** indicating the availability. Shift extends BPSim CalendarParameter [16].

[1] http://bimp.cs.ut.ee
[2] https://www.bizagi.com
[3] https://www.bonitasoft.com
[4] https://www.visual-paradigm.com
[5] https://www.trisotech.com/release-notes/bpmn-modeler

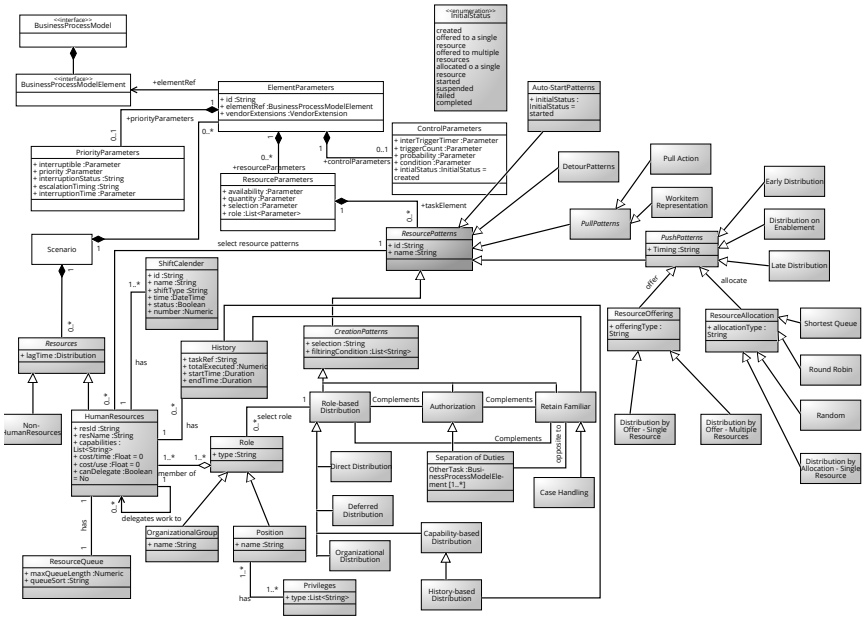

Fig. 1. New extended BPSim MetaModel

ResourcePattern: extends ResourceParameter defined in BPSim and specifies the required resource patterns as discussed in subsection 2.1 that should be applied on ElementParameter. **CreationPatterns:** specifies the requirements to execute a work item. **PushPatterns** include a timing parameter for work item distribution, work item offering to resources based on distribution by offer to single or multiple resources and work item allocation to a single resource based on shortest queue, round robin or random distribution. **PullPatterns** defines the awareness of resources with offered work items required for execution either from direct allocation or a shared work list. It includes two actions: 1. the pull action either from allocation or offering and 2. the sequence of work item representation in the resource waiting queue. **Detour Pattern:** defines the interruptions occurred to work items including escalation, redo, skip, etc. **Auto-Start Patterns** specifies the initial status of work item to *Started* so work item would start immediately. Table 1 identifies the defined classes and parameters in the extended metamodel and their equivalent resource patterns.

ElementParameter was modified in the following way: **1. ControlParameter modification.** based on work item life cycle [13], a state parameter indicating the work item transition from creation to offering then allocation. Work item is then started and moves either to completed, failed or suspended and then resumed. **2. PriorityParameter modification.** a new string attribute indicating the interruption status was added, also the interruption time attribute based on TimingParameter;

Table 1. Extended classes and related resources patterns

Class	**Parameter**	**Pattern Category**	**Pattern**
HumanResource	resName	Creation	Direct distribution
	capabilities	Creation	Capability-based distribution
	canDelegate	Detour	Delegation
ResourceQueue	queueSort	Pull	System-determined work queue content, Resource-determined work queue content
PullAction		Pull	Resource-initiated allocation, Resource-initiated execution- allocated work item, Resource-initiated execution- offered work item
Workitem Representation		Pull	Selection autonomy
Role	type	Creation	Role-based distribution
OrganizationalGroup	name	Creation	Organizational distribution
Privileges	type	Creation	Authorization
History	totalExecuted	Creation	Retain familiar
History	taskRef	Creation	History-based distribution Separation of duties
elementParameter	id (process)	Creation	Case handling
PushPatterns	Timing	Push	Early distribution, Distribution on enablement, Late distribution
ResourceOffering	offeringType	Push	Distribution by offer- single resource, Distribution by offer- multiple resources
ResourceAllocation	allocationType	Push	Distribution by allocation- single resource, Random allocation, Round robin allocation, Shortest queue
Auto-StartPatterns	initialStatus= "Started"	Auto-Start	Commencement on Creation

3 Example

The example explains a simple business process for "Car Maintenance", see Fig. 2. When cars arrive, an administration employee receives and records car information, the selection, offering and allocation of resource are following these patterns: 1. **Creation (selection):** *Role-based Distribution*, 2. **Push:** distribution timing based on *Distribution on Enablement*; offering based on *Distribution by Offer-Multiple Resources* and allocation based on *Round Robin*. The car is then sent to the mechanical department where an engineer is selected based on Role-based Distribution and Capability-based Distribution patterns with experience of 3 years and possession of a certificate. Finally, an accountant receives and records payments for the work done. The accountant pulls work items from his working queue based on work item priority. Listing .1 is an excerpt of the XML for the simulation scenario parameters based on the extended metamodel from Section 2.3

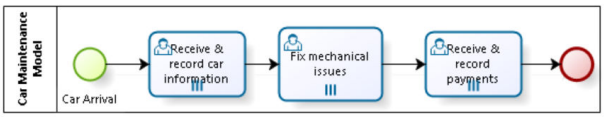

Fig. 2. Simple car maintenance process in BPMN

that realizes the process and patterns discussed above. ElementParameter of "Receive & record car information"task is defined in lines 4 to 13 including the new updates in ControlParameters (initialStatus) and PriorityParameters (escalation-Timing). ResourcePatterns are defined in lines 8 to 12 specifying creation pattern (role-based distribution), push patterns (resource offering, resource allocation and distribution timing). Parameters for the other tasks are defined in lines 13 to 31. Human resources participating in the scenario are defined in lines 32 to 49 specifying resource id, name, capabilities, lagTime, role, organizationalGroup, position, and resource queue properties.

```
1  <bpsim:Scenario id="S1" name="Scenario1:Car Maintenance Model" ...>
2    <bpsim:ScenarioParameters baseTimeUnit="min"/>
3    ...
4    <bpsim:ElementParameters elementRef="Receives & records car information">
5      ...
6      <bpsim:ControlParameters intialStatus="created"/>
7      <bpsim:PriorityParameters interruptible="True" escalationTiming="PT15M"/>
8      <bpsim:ResourcePatterns>
9        <bpsim:Role-basedDistribution roleName="Administration Employee" requiredResQty="1"/>
10       <bpsim:ResourceOffering Timing="Distribution on Enablement" OfferingType="Distribution by Offer-
             Multiple Resources"/>
11         <bpsim:ResourceAllocation AllocationType="round robin"/>
12     </bpsim:ResourcePattern>
13   </bpsim:ElementParameters>
14   <bpsim:ElementParameters elementRef="Fix mechanical issues ">
15       <bpsim:ResourcePattern>
16       <bpsim:Role-basedDistribution roleName="Maintenance Engineer" requiredResQty="1"/>
17       <bpsim:Capability-basedDistribution>
18         <bpsim:Capability experience="3years" certificate="yes"/>
19       </bpsim:Capability-basedDistribution>
20       <bpsim:ResourceOffering Timing="Distribution on Enablement" OfferingType="Distribution by Offer-
             Multiple Resources"/>
21         <bpsim:ResourceAllocation AllocationType="round robin"/>
22       </bpsim:ResourcePattern>
23   </bpsim:ElementParameters>
24   <bpsim:ElementParameters elementRef="Receive & record payments">
25     ...
26     <bpsim:ResourcePatterns>
27       ...
28       <bpsim:PullAction= "Resource-Initiated Execution-Allocated Work Item"/>
29       <bpsim:WorkitemRepresentation= "Resource-Determined Work Queue Content" queueSelection="
             Priority"/>
30     </bpsim:ResourcePatterns>
31   </bpsim:ElementParameters>
32   <bpsim:HumanResources>
33   <bpsim:HumanResource id="Res1" name="Adam" costUnit="$20/h" canDelegate="No">
34         <bpsim:Capabilities experience="3 years" certificate="yes"/>
35         <bpsim:LagTime>
36         <bpsim:UniformDistribution min="3" max="10">
37         </bpsim:LagTime>
38           <bpsim:Role Type="Administration Employee" >
39         <bpsim:OrganizationalGroup name="Administration Department"/>
40             <bpsim:Position name="Reception Employee">
41                 <Privileges/>
42         </bpsim:Position>
43       </bpsim:Role>
44       <bpsim:ShiftCalender/>
45       <bpsim:History/>
46       <bpsim:ResourceQueue maxQueueLength="20" queueSort="Priority"/>
47     </bpsim:HumanResource>
48         ...
49   </bpsim:HumanResources>
50     ...
51 </bpsim:Scenario>
```

Listing .1. RBPSim XML for the example

4 Conclusion and Outlook

In this paper, we have introduced RBPSim, a resource-aware extension for the BPSim metamodel, for better representation of resources within simulation scenarios. The extension is based on workflow resource patterns. RBPSim provides a tool independent exchange format for so-called simulation scenarios including the resources perspective.

In future, we plan to introduce the remaining resource patterns. Moreover, we aim at starting an implementation of the extended BPSim metamodel. The implementation may have two directions. The first is to seek an open source BPS tool that supports BPSim to apply the extended metamodel. The second direction is to implement the extension using a general-purpose simulation tool.

References

1. van der Aalst, W.M.P., van Hee, K.M., ter Hofstede, A.H.M., Sidorova, N., Verbeek, H.M.W., Voorhoeve, M., Wynn, M.T.: Soundness of workflow nets: classification, decidability, and analysis. Formal Asp. Comput. 23(3), 333–363 (2011)
2. van der Aalst, W.M.: Business process simulation revisited. In: Workshop on Enterprise and Organizational Modeling and Simulation. pp. 1–14. Springer (2010)
3. Van der Aalst, W.M., Nakatumba, J., Rozinat, A., Russell, N.: Business process simulation. In: Handbook on BPM 1, pp. 313–338. Springer (2010)
4. Awad, A., Weidlich, M., Weske, M.: Visually specifying compliance rules and explaining their violations for business processes. J. Vis. Lang. Comput. 22(1), 30–55 (2011)
5. Banks, J., Carson, J., Nelson, B., Nicol, D.: Discrete-event system simulation. Prentice Hall, 5th ed. edn. (2010)
6. Freitas, A.P., Pereira, J.L.M.: Process simulation support in bpm tools: The case of bpmn. In: Proceedings of 2100 Projects Association Joint Conferences (2015)
7. Januszczak, J., Hook, G.: Simulation standard for business process management. In: Simulation Conference (WSC), Proceedings of the 2011 Winter. pp. 741–751. IEEE (2011)
8. Laue, R., Müller, C.: The business process simulation standard (bpsim): Chances and limits. In: ECMS. pp. 413–418 (2016)
9. Nakatumba, J., Rozinat, A., Russell, N.: Business process simulation: How to get it right. In: Int. Handbook on Business Process Management. Springer (2009)
10. OMG: Business process model and notation (bpmn version 2.0) (2011)
11. Pufahl, L., Weske, M.: Extensible BPMN process simulator. In: BPM Demo. CEUR Workshop Proceedings, vol. 1920. CEUR-WS.org (2017)
12. Russell, N., van der Aalst, W.M., Ter Hofstede, A.H., Edmond, D.: Workflow resource patterns: Identification, representation and tool support. In: International Conference on Advanced Information Systems Engineering. pp. 216–232. Springer (2005)
13. Russell, N., Ter Hofstede, A.H., Edmond, D., van der Aalst, W.M.: Workflow resource patterns. Tech. rep., BETA Working Paper Series, WP 127, Eindhoven University of Technology, Eindhoven (2004)
14. Waller, A., Clark, M., Enstone, L.: L-sim: Simulating bpmn diagrams with a purpose built engine. In: Simulation Conference. pp. 591–597. IEEE (2006)
15. WfMC: Xml process definition language (xpdl version 2.2) (2012)
16. WfMC: Bpsim - business process simulation specification (2016)

Supporting IoT Application Middleware on Edge and Cloud Infrastructures

Sven Akkermans, Stefanos Peros, Nicolas Small, Wouter Joosen, Danny Hughes

imec-DistriNet, KU Leuven
Celestijnenlaan 200A, 3001, Leuven, Belgium
{first}.{last}@cs.kuleuven.be

Abstract. The Internet-of-Things (IoT) is evolving from classic networks to cloud-style heterogeneous infrastructures, including both edge and cloud entities. As a result, application creators, system managers and infrastructure providers are now distinct stakeholders. The problem is that current IoT middleware solutions are: (1) insufficiently expressive, limiting the specification of application aspects such as Quality-of-Service, and/or (2) require infrastructure-specific knowledge from application creators to meet their application requirements. This paper argues that it is essential for middleware to enable application creators to define functional and non-functional aspects of applications since middleware require this information to correctly deploy and manage applications. Accordingly, expressive abstractions are needed to specify applications as service compositions while hiding the increased complexity of emerging IoT infrastructures. Concretely, the contributions are: (1) a classification of functional and non-functional aspects of IoT applications and (2) requirements for IoT application middleware to simplify application management for different stakeholders.

Keywords: Internet-of-Things, Application Middleware, Cloud Computing

1 Introduction

The Internet-of-Things (IoT) is changing from Wireless Sensor Networks (WSNs) connected with remote cloud servers to cloud-style heterogeneous infrastructures through new paradigms such as edge computing and high-level specification languages. Smart devices are becoming part of a multi-tier IoT infrastructure with both edge (e.g., end-devices, gateways) and cloud entities capable of supporting applications composed out of services [6].

This evolution leads to three distinguishable categories of stakeholders: application creators, system managers and infrastructure providers. Application creators are application-domain experts with little middleware or infrastructure knowledge. System managers provide application creators with a middleware to specify, deploy and manage their applications. Infrastructure providers supply the physical and virtual resources of the IoT infrastructure.

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

This distinction between stakeholders challenges middleware to provide expressive abstractions for specifying functional and non-functional application aspects while hiding infrastructure complexity. To the best of our knowledge, current middleware solutions provide limited support for application runtime specifications or rely on the infrastructure knowledge of the application creator to make deployment decisions [2].

This paper decomposes IoT applications into a generic classification of functional and non-functional aspects as a first step to tackle this challenge. From this classification, we propose requirements for middleware to support expressive and transparent creation, deployment and management of applications on the edge and cloud IoT infrastructure. Modelling IoT applications as compositions of functional and non-functional aspects allows application creators to specify the function and runtime characteristics of their applications and makes it simpler for middleware to match applications to the available resources.

In summary, the contributions of this paper are (1) a classification of generic functional and non-functional aspects of IoT applications, and (2) requirements for middleware to support expressive and transparent creation, deployment and management of IoT applications on edge and cloud infrastructures.

2 Classification of IoT Application Aspects

This section presents a high-level classification of IoT application aspects by extracting application specifications and requirements from related research. We analysed 26 papers from previous EWSN, IPSN and SenSys conferences and 36 papers focused on keywords around WSNs, IoT applications, Edge and Cloud computing[1]. We classify high-level application aspects into two categories, similarly to [1]: functional elements and non-functional properties. Functional elements express pure functionality, e.g., what the functions do. Non-functional properties express runtime properties of the application, e.g., how the functions are performed. The values of their configuration parameters depend on the capabilities of the available resources in the infrastructures and middleware in question. The goal is to provide a representative set of values to guide middleware in using the classification of functional and non-functional application aspects.

Functional Application Elements. Representative functional elements can be obtained from the cross-cut of functional concepts listed in the surveyed papers. Due to space concerns, we limit discussion here to well-cited WSN application papers [4,7,11] and IoT application frameworks [12,14].

Bai et al. [4] focus on sensor networks to develop a taxonomy of WSN archetypes. They extract eight key application concepts: *mobility, initiation of sampling, initiation of data transmission, actuation, interactivity, data interpretation, data aggregation and homogeneity.* **Rahman et al**. [14] survey the literature for IoT frameworks and map functionality to physical devices. They define twelve

[1] Available online at: `goo.gl/Mw9p1p` [Last Access: January 2017]

typical IoT functions: *sense, actuate, profile, device management, control, application, API, discovery, storage, vertical analytics, horizontal analytics and translation*. **Oppermann et al**. [11] survey a decade of WSN applications and present a taxonomy of WSNs. They focus on WSN design aspects for common applications: *goal, sampling approach, sensed phenomenon, data rate, heterogeneity, mobility, connectivity, processing, storage, services and communication primitives*. **Patel et al**. [12] present a development methodology that separates IoT application development into different concerns. They define a sequence of steps for IoT application development, separated into four concerns (i.e., domain, functional, deployment, and platform). They list five concepts that map to software components which encapsulate system functionality: *sensor, storage, computation, actuator and user interface*. **Greenstein et al**. [7] propose a sensor network application construction kit that is based on smart application service libraries. Their library contains six components, which together create applications: *sense, aggregate, transmit, route, process data and storage*.

The analysis of these papers results in six functional elements and associated configuration parameters that are commonly present in IoT application design. Table 1 provides an overview of the functions with sample configuration values. **Data Source** is the function that creates data, e.g., through measurements. Location determines where the function has to be done. Data type dictates the type of the data. Initiation method determines when the function performs. Result transmission governs when the function result is transmitted. **Actuator** is the function that performs physical actions. Action type is the type of performed physical action. **Store** is the function that stores data. Data gathering specifies how the function receives data. Storage processing determines how data is processed before being stored. Storage method governs how the data is stored. **Processor** is the function that processes data. Processing method determines what is done with the data. **Interface** is the function that presents data to the front-end. Data view determines how the data is presented. **Function Manager** is a meta-function that manages other functions. Query functions retrieves metadata from functions. Control functions controls other functions.

Table 1. Functions and configuration parameters with possible values.

Function	**Configuration Parameter**	**Configuration Parameter**
Data Source	Location (Spot, Local, Mobile, ...)	Data Type (Motion, Weather, ...)
	Initiation Method (Event-Driven, Periodic, ...)	Result Transmission (Passive, Active, ...)
Actuator	Location (Spot, Local, Mobile, ...)	Action Type (On/Off, Move, Buzz, ...)
	Initiation Method (Event-Driven, Periodic, ...)	
Store	Data Gathering (Passive, Active, ...)	Storage Processing (Filter, Compress, ...)
	Storage Method (Caching, Persistent, ...)	
Processor	Initiation Method (Event-Driven, Periodic)	Data Gathering (Receive, Request, ...)
	Processing Method (Sort, Trigger, Translate, ...)	Result Transmission (Passive, Active,...)
Interface	Data Gathering (Receive, Request, ...)	Data View (Event-Driven, Periodic, ...)
Function Manager	Query Functions (Discover, Expose, ...)	Control Functions (Configure, Migrate, ...)

Non-Functional Properties. Middleware requires application creators to specify properties during application specification to deploy and manage their applications. The middleware determines the possible specifications of the properties and how to meet them based on its managed infrastructures.

We discuss some identified papers though research in non-functional aware deployment for Cloud/Edge IoT applications is sparse [2]. **Horre et al.** [9] identify quality aware deployment specifications as an important aspect of software deployment for multi-purpose WSNs, but only consider coverage. **Heinzelman et al.** [8] propose a middleware solution that supports QoS for applications on top of WSNs. However, they integrate the QoS application requirements with sensor network management, tightly coupling QoS support with the underlying infrastructure, limiting it to sensor networks and neglecting edge and cloud infrastructures. **Brogi et al.** [5] propose a model to support QoS-aware deployment of IoT applications over Fog infrastructures, but consider only bandwidth and latency as metrics. These papers partially list non-functional metrics to evaluate their models, but lack a classification of non-functional properties for the IoT.

Table 2 lists the extracted non-functional properties, how they can be specified and the main factors of the infrastructure that influence them. The infrastructure providers determine how their infrastructure influences the non-functional property at hand.

Table 2. Non-functional elements, their specifications and main infrastructure factors.

Property	Possible Specification	Infrastructure Factors
Latency	Upper bound in time units	Network
Lifetime	Range in time units or run count	Resource Usage
Dependability	Percentage of uptime	Verification
	Degree of redundancy	Redundancy
Security	Required encryption level	Encryption
	Data placement and movement	Data Locality
Cost	Monetary unit per resource usage or time unit	Pricing Model

Latency is the communication delay between the functions of an application, specified in time units and commonly upper bounded. The main influencing factor is the infrastructure network where the communicating functions reside. The network protocol and topology direct latency in several ways. Influencing factors are routing strategies, hop depth, crossing network boundaries and co-locating entities to avoid communication overhead [10,13]. **Lifetime** of a function refers to the duration (in time units) or the amount of times it performs work [13]. Infrastructure resource usage is the main influencing factor for lifetime. Edge and cloud infrastructures can have battery-powered resources, such as IoT end-devices, or resources with limited usage allowance, such as leasing time on cloud VMs. **Dependability** is the property of a function to perform correctly. It can be expressed as desired uptime or through the degree of redundancy. The infrastructure increases dependability by way of verification, to ensure functions perform as

intended, and redundancy, to protect against failure. Function working verification can be done through heartbeats for uptime and message acknowledgements to ensure packet delivery and redundancy through multiplexing a function, for example, using multiple sensors for more sources. **Security**, as well as privacy, of IoT applications is a broad subject, mostly concerned with the data of the application. Data is more secure and private when encrypted or handled in trusted domains. Therefore, we abstract data placement, movement and its encryption as specifications for privacy and security. Middleware influences this through function placement in the infrastructure. Placing functions on encryption-featured resources and bounding data or data flows to or between trusted resources or areas, avoid unsecured resources having access to the data. **Cost** refers to the cost of the application to run on the infrastructure. Two methods that can be used to express cost requirements of an IoT application, based on research around cloud pricing models [3] are: pay-per-use and subscription-based. In the former, the application is charged based on resource usage while in the latter it is charged on a time basis. Infrastructure determines the pricing model and monetary unit.

3 Requirements for IoT Application Middleware

This section defines requirements and guidelines for IoT application middleware based on the classification proposed in Section 2 and the distinction between stakeholders. System managers need to provide expressive application specifications to application creators which shield them from the underlying infrastructures of the infrastructure providers. This is simplified through blueprints, proposed abstractions of application specifications, essentially a contract from the application creator to the middleware. System managers use blueprints to inform application creation, deployment and management.

Requirements. In this section, we identify four high-level requirements for IoT application middleware. These requirements are defined with respect to the needs of the application creator and the required support of the underlying middleware:

- **R1:** Application creators should be able to specify *what* an application does [1]. Middleware should provide the functions that are possible on the infrastructures to compose applications.
- **R2:** Application creators should be able to specify *how* an application operates [1]. Middleware should provide the ability to specify the non-functional properties it can support.
- **R3:** Application creators have their applications deployed and managed transparently to the underlying infrastructures. Middleware should autonomously insert additional functions if necessary in the infrastructure to support function interoperability and non-functional properties.
- **R4:** Application creators should only be able to specify functions and non-functional properties that are possible in the provided infrastructures. Middleware should maintain a consistent view of the capabilities of its resources and infrastructures and relate them to functions and non-functional properties.

3.1 Blueprints

In this section, we propose *blueprints*, an application specification abstraction based on the classification in Section 2 and the requirements in Section 3. Three distinct elements can be derived from **R1**, **R2** and **R3** for application specification: user functions, middleware functions and non-functional properties. Application creators compose blueprints through a combination of user functions, essentially services, and non-functional properties. The middleware, with a view of the resources and infrastructures, as specified in **R4**, provides suitable options to application creators when creating blueprints. For instance, the middleware should enable creating a Data Source function with a mobile location option on a mobile tracking resource in the infrastructure. User functions are specified by the application creator while middleware functions are autonomously inserted by the middleware. Middleware presents the possible functions, configuration parameters and values to the application creators, which they specify and connect. Middleware functions can be necessary to support non-functional properties or other functions, depending on the infrastructure, and so should be autonomously inserted by the middleware. For instance, communicating functions on different resources may need an intermediate Processor function for translation or functions might need to be periodically exposed by a Function Manager to ensure reliability.

Blueprints support stakeholders in three phases of the application life-cycle. For creation, blueprints remove the burden of programming from application creators by providing generic functions and allow expressing non-functional properties. This is motivated by basing functions and properties on a wide range of existing research. For deployment, blueprints provide the middleware with the application requirements it needs to intelligently deploy the application on the suitable resources. For management, blueprints allow to balance changing infrastructural concerns with ongoing application requirements. For instance, in case of unexpected resource failures, replacement functions can be inserted.

4 Discussion and Conclusion

This work supports middleware in the management of IoT applications across multi-tier infrastructures. IoT middleware can implement blueprints by supporting the listed functions, essentially services, and non-functional properties on their managed infrastructure. This requires middleware to have suitable configurable programming constructs (e.g., Python scripts, Contiki ELF modules, ...) for functions and the ability to define non-functional properties on top of them. For instance, if the middleware supports application specification in an XML-language (e.g., as in [9]), the discussed functions and properties should be part of the language the application creators use. The resulting application specification is a blueprint and enables the middleware to deploy and manage the application. In practice, blueprints are currently being implemented on Niflheim, an end-to-end middleware for applications on a multi-tier IoT infrastructure [15].

This paper contributes a classification of functional elements and non-functional properties, essential for application specification, and a set of requirements for

middleware to support the management of IoT applications, transparently to the emerging edge and cloud infrastructures. This research is based on existing literature which covers a wide range of applications. To conclude, this work supports IoT application middleware in application specification through blueprints and by listing necessary requirements to foster application use on IoT infrastructures.

References

1. Aldinucci, M., Danelutto, M., Kilpatrick, P.: Autonomic management of non-functional concerns in distributed parallel application programming. In: 2009 IEEE International Symposium on Parallel Distributed Processing. pp. 1–12
2. Arcangeli, J.P., Boujbel, R., Leriche, S.: Automatic deployment of distributed software systems: Definitions and state of the art. Journal of Systems and Software 103, 198–218 (2015)
3. Artan Mazrekaj, I.S., Sejdiu, B.: Pricing schemes in cloud computing: An overview. International Journal of Advanced Computer Science and Applications 7(2) (2016)
4. Bai, L.S., Dick, R.P., Dinda, P.A.: Archetype-based design: Sensor network programming for application experts, not just programming experts. In: 2009 International Conference on Information Processing in Sensor Networks. pp. 85–96 (Apr 2009)
5. Brogi, A., Forti, S.: Qos-aware deployment of iot applications through the fog. IEEE Internet of Things Journal (2017)
6. Chiang, M., Zhang, T.: Fog and IoT: An Overview of Research Opportunities. IEEE Internet of Things Journal 3(6), 854–864 (Dec 2016)
7. Greenstein, B., Kohler, E., Estrin, D.: A Sensor Network Application Construction Kit (SNACK). In: Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems. pp. 69–80. SenSys '04, ACM
8. Heinzelman, W.B., Murphy, A.L., Carvalho, H.S., Perillo, M.A.: Middleware to support sensor network applications. IEEE network 18(1), 6–14 (2004)
9. Horre, W., Michiels, S., Joosen, W., Hughes, D.: QARI: Quality Aware Software Deployment for Wireless Sensor Networks. pp. 642–647. IEEE (2010)
10. Mahmud, R., Buyya, R.: Fog computing: A taxonomy, survey and future directions. CoRR abs/1611.05539 (2016)
11. Oppermann, F.J., Boano, C.A., Romer, K.: A Decade of Wireless Sensing Applications: Survey and Taxonomy. In: Ammari, H.M. (ed.) The Art of Wireless Sensor Networks, pp. 11–50. Springer Berlin Heidelberg, Berlin, Heidelberg (2014)
12. Patel, P., Cassou, D.: Enabling high-level application development for the Internet of Things. Journal of Systems and Software 103(Supplement C), 62–84 (May 2015)
13. Priyantha, N.B., Kansal, A., Goraczko, M., Zhao, F.: Tiny web services: Design and implementation of interoperable and evolvable sensor networks. In: Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems. pp. 253–266
14. Rahman, L.F., Ozcelebi, T., Lukkien, J.J.: Choosing Your IoT Programming Framework: Architectural Aspects. In: 2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud). pp. 293–300 (Aug 2016)
15. Small, N., Akkermans, S., Joosen, W., Hughes, D.: Niflheim: An end-to-end middleware for applications on a multi-tier iot infrastructure. In: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA). pp. 1–8

Towards Decentralized Auto-Scaling Policies for Data Stream Processing Applications

Gabriele Russo Russo

Department of Civil Engineering and Computer Science Engineering
University of Rome Tor Vergata, Italy
russo.russo@ing.uniroma2.it

Abstract Data Stream Processing applications can process large data volumes in near real-time. In order to face varying workloads in a scalable and cost-effective manner, it is critical to adjust the application parallelism at run-time. We formulate the elasticity problem as a Markov Decision Process (MDP). As the MDP resolution requires full knowledge of the system dynamics, which is rarely available, we rely on model based Reinforcement Learning to improve the scaling policy at run-time. We show promising results even for a decentralized approach, compared to the optimal MDP solution.

Keywords: Data Stream Processing, Elasticity, Reinforcement Learning

1 Introduction

New emerging application scenarios (e.g., social analytics, fraud detection, Smart City) leverage Data Stream Processing (DSP) to process data streams in near real-time. A DSP application is usually represented as a directed acyclic graph (DAG), with data sources and operators as vertices, and streams as edges [5]. Each *operator* continuously receives data (e.g, *tuples*), applies a transformation, and generates new outgoing streams.

A commonly adopted DSP optimization is *data parallelism*, which consists of scaling-in/out the parallel instances of the operators, so that each processes a portion of the incoming data (at the cost of using more resources) [5]. Due to the unpredictable and variable rate at which the sources produce the streams, a key feature for DSP systems is the capability of *elastically* adjusting the parallelism at run-time. Most of the existing DSP frameworks allow to allocate more than one replica per operator, but their support for the run-time reconfiguration is quite limited, as regards both the mechanisms and the policies.

In this paper, we focus on the auto-scaling policies. We formalize the elasticity problem for a DSP application as a Markov Decision Process (MDP), presenting both centralized and decentralized formulations. Unfortunately, in practice the optimal MDP policy cannot be determined, because several system dynamics may be unknown. To cope with the model uncertainty, we rely on Reinforcement Learning (RL) approaches, which learn the optimal MDP policy on-line by

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

interacting with the system. Specifically, we present a model based RL solution that leverages the partial knowledge of the model to speedup the learning process.

Elasticity for DSP is attracting many research efforts [1], with most approaches relying on heuristics to determine the scaling decisions. An optimization model that also considers the operator placement problem has been presented in [2], but it cannot be easily solved in a decentralized manner. Here we describe a simpler model, for which we can derive a decentralized formulation. The application of RL techniques to DSP elasticity is quite limited. Heinze et al. [4] propose a simple RL approach to control the system utilization, but they focus on infrastructure-level elasticity. Lombardi et al. [6] exploit RL in their elasticity framework as well, but the learning algorithm is only used for thresholds tuning. In [3] different RL algorithms have been compared for solving the elasticity problem for a single DSP operator in isolation, while in this work we consider whole applications.

In the rest of this paper, we first formulate the elasticity problem as an MDP in Sect. 2, presenting in Sect. 3 the RL based algorithm for learning the scaling policy; we evaluate the proposed solutions in Sect. 4, and conclude in Sect. 5.

2 Problem Formulation

In this paper, we consider the elasticity problem for a DSP application composed of N operators. Each operator is possibly replicated into a number of instances and, without lack of generality, we assume even distribution of the incoming data among the parallel instances. For each operator, an *Operator Manager* monitors the operator functionality, while an *Application Manager* supervises the whole application. The number of parallel instances used by each operator is adjusted either by its Operator Manager (*decentralized* adaptation) or by the Application Manager (*centralized* adaptation).

At each decision step, for each operator we can add an instance, terminate one, or keep the current parallelism. Following a scaling decision, the operator is subject to a reconfiguration process; as the integrity of the streams and the operator internal state must be preserved, the whole application is usually paused during the process, leading to *downtime* [2]. Our goal is to take reconfiguration decisions as to minimize a long-term cost function which accounts for the downtime and for the monetary cost to run the application. The latter comprises (i) the cost of the instances allocated for the next decision period, and (ii) a penalty in case of a Service Level Agreement (SLA) violation. In particular, we consider a constraint on the application response time[1], so that a penalty is paid every time the response time exceeds a given threshold.

In order to keep the system model simple, we consider a deployment scenario with (i) homogeneous computing resources on which the operator instances are executed, and (ii) negligible communication latency between them. We defer to future work the extension of the model for a more realistic distributed setting.

[1] We define the response time as the maximal source-sink total processing latency over the application DAG.

System Model In the considered system, reconfiguration decisions are taken periodically. Therefore, we consider a slotted time system with fixed-length intervals of length Δt, with the i-th time slot corresponding to the time interval $[i\Delta t, (i+1)\Delta t]$. We denote by $k_{op,i} \in [1, K_{max}]$ the number of parallel instances at the beginning of slot i for operator op, and by $\lambda_{op,i}$ its average input rate measured during the previous slot. Additionally, we use Λ_i to denote the overall application input rate (i.e., the total data sources emission rate). At the beginning of slot i, a decision a_i is made on whether reconfiguring each operator.

We first consider a centralized model, in which the reconfiguration decisions are taken by the Application Manager; then, at the end of the section, we consider the case in which the responsibility of making scaling decisions is decentralized, and each Operator Manager acts as an independent agent. In both the cases, we formalize the resulting problem as a discrete-time Markov Decision Process (MDP).

Centralized Elasticity Problem An MDP is defined by a 5-tuple $\langle \mathcal{S}, \mathcal{A}, p, c, \gamma \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}(s)$ a finite set of actions for each state s, $p(s'|s,a)$ are the state transition probabilities, $c(s,a)$ is the cost when action a is executed in state s, and $\gamma \in [0,1]$ is a future cost discounting factor.

We define the state of the system at time i as $s_i = (\Lambda_i, k_{1,i}, k_{2,i}, \ldots, k_{N,i})$. For the sake of analysis, we discretize the arrival rate Λ_i by assuming that $\Lambda_i \in \{0, \bar{\Lambda}, \ldots, L\bar{\Lambda}\}$ where $\bar{\Lambda}$ is a suitable quantum. For each state s, the action set is $\mathcal{A}(s) = \mathcal{A}_1(s) \times \cdots \times \mathcal{A}_N(s)$, where, for each operator op, $\mathcal{A}_{op}(s) = \{+1, -1, 0\}$ (except for the boundary cases with minimum or maximum replication).

System state transitions occur as a consequence of scaling decisions and arrival rate variations. It is easy to realize that the system dynamic comprises a stochastic component due to the exogenous rate variation, and a deterministic component due to the fact that, given action a and the current number of instances, we can readily determine the next number of instances. An example of a system state transition is illustrated in Fig. 1.

To each state pair we associate a cost $c(s,a)$ that captures the cost of operating the system in state s and carrying out action a, including:

1. the *resource* cost $c_{res}(s,a)$, required for running $(k_{op} + a_{op})$ instances for each operator op, assuming a fixed cost per instance;
2. the reconfiguration cost $c_{rcf}(a)$, which accounts for the application downtime, assuming a constant reconfiguration penalty;
3. the SLA violation cost $c_{SLA}(s,a)$, which captures the penalty incurred whenever the response time $T(s,a)$ violates the threshold T_{SLA}.

We define the cost function $c(s,a)$ as the weighted sum of the normalized terms:

$$c(s,a) = w_{res} \frac{\sum_{o=1}^{N} k_{op} + a_{op}}{NK_{max}} + w_{rcf} \mathbb{1}_{\{\exists o: a_o \neq 0\}} + w_{SLA} \mathbb{1}_{\{T(s,a) > T_{SLA}\}} \quad (1)$$

where w_{res}, w_{rcf} and w_{SLA}, $w_{res} + w_{rcf} + w_{SLA} = 1$, are non negative weights.

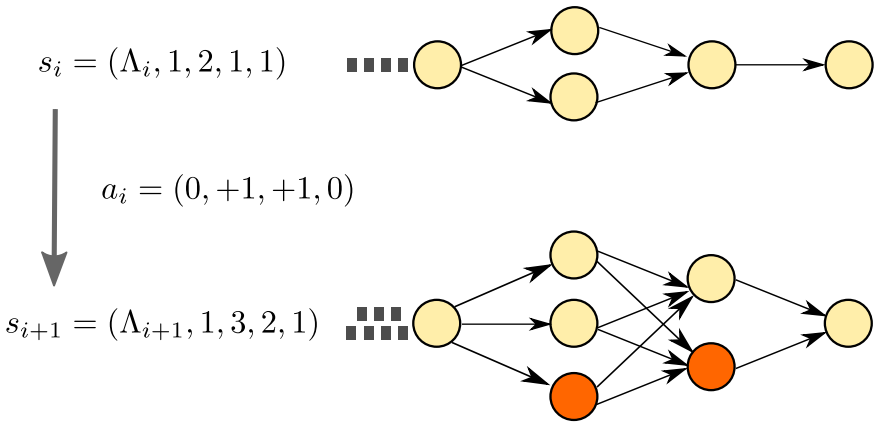

Figure 1: Example of a state transition in the centralized MDP model. At time i, the application input rate is Λ_i and the components run a single instance, except for the second one which run two. The ApplicationManager picks action $(0, +1, +1, 0)$ at time i, thus adding an instance of the second and the third operator. The resulting parallelism degree of the operators at time $i + 1$ is, respectively, 1, 3, 2, and 1. The input rate at time $i + 1$ is Λ_{i+1}, which obviously does not depend on a_i.

Decentralized Elasticity Problem In the decentralized adaptation scenario, we assume that each Operator Manager independently acts on its associated operator, having only a local view of the system. We again rely on MDP to formalize the cost minimization problem for each agent (i.e., the Operator Managers). Omitting the reference to the specific operator, we define the state at time i as the pair $s_i = (\lambda_i, k_i)$, where λ_i is discretized using a suitable quantum for each operator. The action set is simply $\mathcal{A}(s) = \{+1, -1, 0\}$ (except for the boundary cases with minimum or maximum replication).

Because the agents have not a global view of the application, they can only optimize local metrics, and thus we have to formulate a new local cost function $c'(s, a)$. We replace the SLA violation penalty with one based on the operator utilization $U(s, a)$ and a target utilization $\bar{U}$. We get:

$$c'(s,a) = w_{res}\frac{k+a}{K_{max}} + w_{rcf}\mathbb{1}_{\{a\neq 0\}} + w_{util}\mathbb{1}_{\{U(s,a)>\bar{U}\}} \tag{2}$$

where w_{res}, w_{rcf} and w_{util}, $w_{res} + w_{rcf} + w_{util} = 1$, are non negative weights.

3 Learning an Optimal Policy

A policy is a function π that associates each state s with the action a to choose. For a given policy π, let $V^{\pi}(s)$ be the value function, i.e., the expected infinite-horizon

discounted cost starting from s. It is also convenient to define the action-value function $Q^\pi : \mathcal{S} \times A \to \Re$ which is the expected discounted cost achieved by taking action a in state s and then following the policy π:

$$Q^\pi(s,a) = c(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^\pi(s'),\ \forall s \in \mathcal{S} \tag{3}$$

It is easy to realize that the value function V and the Q function are closely related in that $V^\pi(s') = \min_{a \in A(s)} Q^\pi(s',a),\ \forall s \in \mathcal{S}$. More importantly, the knowledge of the Q function is fundamental in that it directly provides the associated policy: for a given function Q, the corresponding policy is $\pi(s) = \arg\min_{a \in A(s)} Q(s,a),\ \forall s \in \mathcal{S}$. We search for the optimal MDP policy π^*, which satisfies the Bellman optimality equation:

$$V^{\pi^*}(s) = \min_{a \in \mathcal{A}(s)} \left\{ c(s,a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s,a) V^{\pi^*}(s') \right\},\ \forall s \in \mathcal{S} \tag{4}$$

In the ideal situation, we have full knowledge of the system, and we can directly compute π^* using the *Value Iteration* algorithm [7]. In more realistic cases, we have only partial knowledge of the underlying system model (e.g., the workload distribution is usually unknown). We can resort to Reinforcement Learning (RL) approaches, which are characterized by the basic principle of learning the optimal policy by direct interaction with the system. In particular, we consider a model based RL algorithm that, at each time step, improves its estimates of the unknown system parameters, and performs an iteration of the Value Iteration algorithm (see Algorithm 1). Simpler model-free RL algorithms like *Q-learning* have been shown to achieve bad performance even on smaller tasks [3].

Algorithm 1 RL based Elastic Control Algorithm

```
1: Initialize the action-value function Q
2: loop
3:     choose an action a_i (based on current estimates of Q)
4:     observe the next state s_{i+1} and the incurred cost c_i
5:     update the unknown system parameters estimates
6:     for all s ∈ S do
7:         for all a ∈ A(s) do
8:             Q_i(s,a) ← ĉ_i(s,a) + γ Σ_{s'∈S} p̂(s'|s,a) min_{a'∈A(s')} Q_{i-1}(s',a')
9:         end for
10:    end for
11: end loop
```

We first consider the case in which the operator response time model is known, and let the algorithm learn the state transition probabilities.. In order to estimate $p(s'|s,a)$, it suffices to estimate the input rate transition probabilities

$P[\lambda_{i+1} = \lambda' | \lambda_i = \lambda]$[2], since the dynamics related to the number of instances are known and deterministic. Hereafter, since λ takes value in a discrete set, we will write $P_{j,j'} = P[\lambda_{i+1} = j'\bar{\lambda} | \lambda_i = j\bar{\lambda}]$, $j, j' \in \{0, \dots, L\}$ for short. Let $n_{i,jj'}$ be the number of times the arrival rate changes from state $j\bar{\lambda}$ to $j'\bar{\lambda}$, in the interval $\{1, \dots, i\}$, $j, j' \in \{1, \dots, L\}$. At time i the transition probabilities estimates are

$$\widehat{P_{j,j'}} = \frac{n_{i,jj'}}{\sum_{l=0}^{L} n_{i,jl}} \tag{5}$$

If we remove the assumption on the known response time model, we have to estimate the cost $c(s, a)$ as well, because we cannot predict the SLA/utilization violation any more. So, we split $c(s, a)$ and $c'(s, a)$, respectively defined in (1) and (2), into known and unknown terms: the known term $c_k(s, a)$ accounts for the reconfiguration cost and the resources cost, whereas the unknown cost $c_u(s, a)$ represents the SLA (or utilization) violation penalty. We use a simple exponential weighted average for estimating the unknown cost:

$$\hat{c}_{u,i}(s, a) \leftarrow (1 - \alpha)\hat{c}_{u,i-1}(s, a) + \alpha c_{u,i} \tag{6}$$

where $c_i, u = w_{SLA}$ (or w_{util}) if a violation occured a time i and 0 otherwise.

As regards the complexity of the algorithm, the size of the state-action space is critical, since each learning iteration requires $O(|\mathcal{S}|^2|\mathcal{A}|^2)$ operations. We observe that in the centralized model $|\mathcal{S}|$ and $|\mathcal{A}|$ grow exponentially with the number of operators N, whereas they are not influenced by N in the decentralized model.

4 Evaluation

We evaluate by simulation the presented models, and compare the policies learned through RL to the MDP optimal one. In order to explicitly solve the MDP, we need a state transition probability matrix, which is not available in practical scenarios. Thus, for evaluation, we consider a dataset made available by Chris Whong[3] that contains information about taxis activity, and extract a state transition probability applying (5). We then evaluate the proposed solutions on a new workload, generated according to those probabilities.

For simplicity, we consider a *pipeline* application, composed of a data source and up to 4 operators. Each operator runs at most $K_{max} = 5$ instances, each behaving as a M/D/1 queue with service rate μ_{op}. For evaluation, we consider a scenario with slightly different service rates, and set $\mu_1 = 3.7$, $\mu_2 = \mu_4 = 3.3$, and $\mu_3 = 2.7$ tuple/s. Because of space limitation, we defer the evaluation of real world topologies to future work. We consider $\Delta t = 1$ min, and aggregate the events in the dataset over one minute windows. We assume $\Lambda_i = \lambda_{o,i}, \forall o$, discretized with $\bar{\lambda} = 20$ tuple/min. For the cost function, we set $w_{sla} = w_{util} = w_{rcf} = 0.4$, $w_{res} = 0.2$, $T_{SLA} = 650$ ms, and $\bar{U} \in \{0.6, 0.7, 0.8\}$. As regards the learning

[2] To simplify notation, we simply use λ to denote the input rate. In the centralized model, we use the same estimates for the total application input rate Λ.

[3] http://chriswhong.com/open-data/foil_nyc_taxi/

Table 1: Results for a 3-operators application. The "+" sign denotes the knowledge of the response time model.

Scaling Policy	Avg. Cost	SLA Violations	Reconf.	Avg. Instances
Centralized MDP	0.163	8903	13882	10.95
Centralized RL$^+$	0.164	9505	13684	10.94
Centralized RL	0.167	15681	14579	10.79
Decentralized ($\bar{U} = 0.6$)	0.178	3639	30104	11.46
Decentralized ($\bar{U} = 0.7$)	0.173	17111	30404	10.30
Decentralized ($\bar{U} = 0.8$)	0.205	79670	29681	9.15

algorithm, we set $\gamma = 0.99$, and $\alpha = 0.1$. We compare the results obtained by solving the MDP to those achieved by the centralized RL algorithm (with and without the known response time model) and by the decentralized solution.

In Table 1 we report the results for a 3-operators topology. As expected, the minimum average cost is achieved solving the MDP; interestingly, the centralized RL solution incurs almost negligible performance degradation, and the gap with the decentralized approach is not significant as well. However, we note that the performance of the decentralized solution depends on the target utilization $\bar{U}$, which has still to be set manually in our work. Setting a too high (or too low) value results in a different trade-off between SLA violations and used instances, with negative effects on the overall cost as well. The decentralized solution shows a higher number of reconfigurations, due to the lack of coordination between the agents. As illustrated in Fig. 2a, the convergence velocity of the different solutions is similar, except for the centralized RL algorithm. In absence of the response time model, the algorithm is indeed significantly slower to learn than the other solutions. When the response time model is known, the algorithm converges much faster, despite the large state-action space.

We also compare the decentralized approach to the MDP varying the number of operators in the application. As shown in Fig. 2b, the cost gap between the two solutions slightly increases as the application gets more complex. However, we observe that the decentralized algorithm has not scalability issues as the number of operators increases, while solving a centralized problem gets easily impractical.

5 Conclusion

In this paper we have formalized the elasticity problem for DSP applications as a Markov Decision Process, and proposed a Reinforcement Learning based solution to cope with the limited knowledge of the system model. Our numerical evaluation shows promising results even for a fully decentralized solution which, leveraging the available knowledge about the system, does not suffer from the extremely slow convergence of model-free RL algorithms. In practical scenarios, we could also combine the proposed solution with a simple threshold-based policy

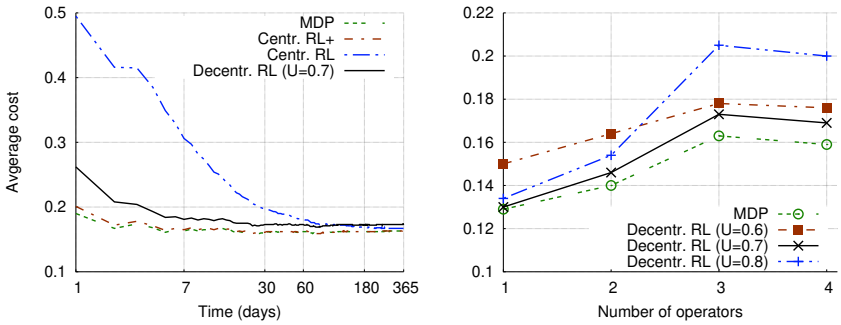

Figure 2: Average cost during one simulated year for a 3-operators application (a), and for different number of operators (b). The "+" sign denotes the knowledge of the response time model.

to be used at the beginning, while the agents learn a good policy to be adopted in the following.

For future work, our goal is twofold. We plan to improve the decentralized learning algorithm exploring RL techniques specifically targeted to multi-agent systems. At the same time, we will extend the model to cope with a more complex and realistic scenario, considering, e.g., resource heterogeneity and network latency in distributed deployments.

References

1. Assuncao, M.D., da Silva Veith, A., Buyya, R.: Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. Journal of Network and Computer Applications 103, 1 – 17 (2018)
2. Cardellini, V., Lo Presti, F., Nardelli, M., Russo Russo, G.: Optimal operator deployment and replication for elastic distributed data stream processing. Concurrency and Computation: Practice and Experience (2017), http://dx.doi.org/10.1002/cpe.4334
3. Cardellini, V., Lo Presti, F., Nardelli, M., Russo Russo, G.: Auto-scaling in data stream processing applications: A model based reinforcement learning approach. In: Proc. of InfQ '17 (in conjunction with VALUETOOLS '17) (2018, to appear)
4. Heinze, T., Pappalardo, V., Jerzak, Z., Fetzer, C.: Auto-scaling techniques for elastic data stream processing. In: Proc. IEEE ICDEW '14. pp. 296–302 (2014)
5. Hirzel, M., Soulé, R., Schneider, S., Gedik, B., Grimm, R.: A catalog of stream processing optimizations. ACM Comput. Surv. 46(4) (Mar 2014)
6. Lombardi, F., Aniello, L., Bonomi, S., Querzoni, L.: Elastic symbiotic scaling of operators and resources in stream processing systems. IEEE Transactions on Parallel and Distributed Systems PP(99), 1–1 (2017)
7. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press, Cambridge (1998)

Markdown Architectural Decision Records: Format and Tool Support

Oliver Kopp[1], Anita Armbruster[1], and Olaf Zimmermann[2]

[1] Institute for Parallel and Distributed Systems, University of Stuttgart
Stuttgart, Germany
{lastname}@ipvs.uni-stuttgart.de
[2] Institute for Software, Hochschule für Technik (HSR FHO)
Rapperswil, Switzerland
olaf.zimmermann@hsr.ch

Abstract. Architectural decision records answer "why" questions about designs and make tacit knowledge explicit. Many architectural decisions are made during development iterations because they have a close connection to the code. It is challenging to come up with task-specific decision capturing practices and supporting tools that are not perceived as time wasters; context switches and media breaks that harm the productivity of coding architects and developers involved in the decision making have to be avoided. To integrate existing architect-centric approaches into the developer toolchain, this paper presents a Markdown-based decision capturing template that we derived from previous work to enhance an existing decision capturing tool for developers. Our early validation results in the context of an open source project suggest that format and tool promise to contribute to an integrated decision capturing practice, with further enhancements being required. Tool and template are available in public GitHub repositories.

1 Introduction

Source code needs to be documented. This typically leads to comments in code and to external documents. Well-written classes and methods, which have expressive names and understandable branches [6], make low-level code comments obsolete. On the other end of the spectrum, however, wide-ranging decisions of high architectural significance are made during development iterations; these decisions are not self-explanatory and not expressed in the code explicitly. An example of such an *architectural decision* is how to keep user session data consistent and current across Web shop instances. Typically, these kind of decisions are recorded in external documentation files, wikis, or tools [18]. The primary tool of developers, however, is their Integrated Development Environment (IDE) with integrated version control system support. Requiring developers to use more tools has a negative impact on productivity, quality, and motivation to capture architecturally significant decisions: opening another tool requires some setup and training effort and leads to undesired, time consuming context switches. Furthermore, model- and document-centric tools typically do not integrate themselves well in the developer's toolchain: The documents are not committed in plain text format into the version control system—if versioned along the code at all. Further, the documents might get out of sync with the

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

code base [13]. As a consequence, architectural decision capturing practices and tools oftentimes are perceived as chronophages (time wasters). This holds true particularly for agile and lean developer communities. We therefore can derive the following problem statement:

How to seamlessly integrate architectural decision making into developer tool landscapes — so that decision rationale can be collected under high coding velocity?

To describe how we propose to overcome this problem, we first provide some background and related work (Sect. 2). We then introduce the *Markdown Architectural Decision Records* (MADR) format as the conceptual contribution of this paper (Sect. 3). Next, we present a tool implementation for MADR integration that makes our conceptual solution available to practitioners and validates the novel format (Sect. 4). We have further validated the MADR format and tooling in an action research (Sect. 5). A discussion on the findings of MADR follows in Sect. 6. Finally, we conclude the paper (Sect. 7).

2 Background and Related Work

A large body of research work on capturing architectural decisions exists; the state of the art is for instance surveyed by Alexeeva et al. [2] and by Capilla et al. [4]. Specifically to the context of service orientation and service composition, the SOA Decision Modeling (SOAD) project [17] investigated architectural decisions recurring in SOA design and introduced a seven-step method to identify, make, and capture such decisions. SOAD used a fixed, rather comprehensive meta model. Taking that experience into account, our template and tool, to be introduced in Sect. 3 and 4, are designed in such as way that they are applicable on service-oriented middleware and tool development projects (as evidenced in the validation activity presented in Sect. 5), but not limited to such projects.

More recently, templates, practices, and tools specifically targeting coding architects and agile developers, who make certain architectural decisions and contribute to others, have been proposed. Seven different formats, including comprehensive and lean ones, are compared by Zimmermann et al. [18]. They also introduce ADMentor, a decision modeling add-in for Sparx Enterprise Architect. ADMentor supports two primary user stories and themes, a) problem space modeling and b) architectural decision capturing. Problem spaces model recurring decisions along with options to be considered. The architectural decision capturing capability then allows architects on projects to keep track of decisions made in a *decision log* as suggested by the ISO/IEC/IEEE 42010 standard [7] for architecture description. Other features include rich text editing, model linking and refactoring, and reporting/analysis. Decision capturing is streamlined by lightweight decision capturing templates such as *Y-Statements* [15]; Question, Option, Criteria (QOC) diagrams [9] are supported as well.

General best practice recommendations for decision documentation are presented by Zdun et al. [15], including the above mentioned Y-Statement format originally developed for—and first applied in–an industry project setting at ABB [16]. Y-Statements contain the aspects context, concern, the chosen option, other possible options, the expected positive results, and the accepted downsides as well as (optionally) additional decision

```
In the context of <use case/user story u>,
facing <concern c>
we decided for <option o>
and neglected <other options>,
to achieve <system qualities/desired consequences>,
accepting <downside / undesired consequences>,
because <additional rationale>.
```

Fig. 1. (WH)Y-Statement format: context and concern form the diagonal two lines at the top of the Y, the other five form the vertical bar of it.

rationale (Fig. 1). As a structured text format, Y-Statements can be put in external documentation, in code comments [6], or in Java annotations[3].

An example of such a Y-statement is: "In the context of the Web shop service, facing the need to keep user session data consistent and current across shop instances, we decided for the Database Session State pattern [5] (and against Client Session State [5] or Server Session State [5]) to achieve cloud elasticity, accepting that a session database needs to be designed, implemented, and replicated."

A rather popular[4] practitioner's tool is `adr-tools`[5]. It uses the format by Nygard [10], which covers less aspects than the Y-Statements. For instance, the neglected options are not shown. Both Y-statements and Nygard's Architecture Decision Records (ADRs) have been designed with a lean and agile mindset that is in line with the vision of software specification and documentation in Continuous Software Development (CSD) [14].

3 Markdown Architectural Decision Records (MADR)

To keep the architectural decisions close to common developer tools and artifacts, we propose to 1) use Markdown as decision capturing syntax (with a proposed format derived from Y-Statements) and 2) place the decisions in the folder `docs/adr` of code projects that are version-controlled.

Markdown is a text format, which enables common version control systems such as git to be used. This makes diffing within the IDE possible. Our decision to use Markdown as markup language (instead of other markup languages) is supported by the following rationale: 1) it eases writing, 2) Markdown is the markup language for comments by users within GitHub (such as in gists, issues, or pull requests), and 3) already available rendering tools can be leveraged.

We call the new format *Markdown Architectural Decision Records (MADR)*. Some early adopters of the Y-Statement syntax had commented that the sentences can get really long and are therefore hard to read for inexperienced readerships. As Markdown is a *structured* text format in which headings can mark sections, we decided to deviate from the pure Y-format and transferred it into a section-oriented one (similar to the successful approach of the `adr-tools` outlined in Sect. 2). The starting point was the "Decision Capture Template" [12], which we adapted to contain all elements of the Y-Statements.

[3] GitHub project "Embedded Architectural Decision Records," `https://adr.github.io/e-adr/`

[4] 630 stars on GitHub as of 2018-01-31

[5] `https://github.com/npryce/adr-tools`

We indicate the backward mapping to the Y-format when describing the new template below (in parenthesis).

Figure 2 shows the format of MADR. Each record takes a title (line 1) followed by the user story (line 3). The user story is made optional because its content further elaborates on the mandatory context (from Y-Statement syntax) and problem statement (line 5). More information on the context such as forces or decision drivers (Y-Statement's concerns and aims) can be appended (line 6). The considered alternatives (including the chosen and the neglected ones; Y-Statement) are listed as items (lines 8 to 11). The chosen alternative includes a justification (Y-Statement's "to achieve" rationale) and optionally consequences (Y-Statement's "accepting that" downsides). Follow-up decisions are are also listed as items (lines 13 to 19). If a longer pro/con evaluation of the alternatives makes sense, each option can be listed as a separate heading followed by an itemized list of pros and cons. In summary, all aspects of a Y-Statement are covered in the template, even though the consequences are left optional.

Note that MADR does not restrict the Markdown syntax. Thus, it is possible to include images, ASCII art, and PlantUML[6].

The folder `docs/adr` was chosen to enable rendering in GitHub pages. Since 2016, GitHub pages allows for rendering a homepage out of the `docs` folder [8]. When updating files in the `docs` folder, GitHub processes them using the Jekyll site generator[7], which basically converts markdown files into HTML files using a given template. As a consequence, when placing the ADRs into a subfolder, it is possible to make them accessible on the World-Wide Web.

4 Tool Implementation and Integration

To support MADR we extended `adr-tools` (made available at `https://github.com/adr/adr-tools`) and created `adr-log` (made available as npm package at `https://www.npmjs.com/package/adr-log`).

The original `adr-tools` support arbitrary formats when creating new architectural decisions by providing an appropriate `template.md` file. New ADRs are put in the format `nnnn - title-lowercased-with-dashes.md` in the directory, where `nnnn` is a number starting from 0001. Besides basic creation functionality, `adr-tools` allows for linking ADRs. For instance, a new ADR can supersede an existing ADR. For that the status of an ADR is tracked under a new heading "Status". In MADR, we record the status and the status changes in a table with the columns "Date" and "Status". We extended `adr-tools` to support the command `adr new docs/adr madr`, where `docs/adr` is the directory where architectural decisions are put and `madr` denotes that MADR should be used as template format. At each call of `adr new TITLE`, MADR is used as template instead of Nygard's template. Furthermore, we are working on supporting the status table in the beginning so that it is 1) created when an ADR is superseded by another ADR (e.g., `adr -s 1 Use SQL Database`, tells `adr-tools` to add a note at ADR-0001 that it is superseded) and 2) amended when there is a new link to an ADR

[6] `http://plantuml.com/`

[7] `https://help.github.com/articles/using-jekyll-as-a-static-site-generator-with-github-pages/`

```
# *[short title of solved problem and solution]*

**User Story:** *[ticket/issue-number]* <!-- optional -->

*[context and problem statement]*
*[decision drivers | forces]* <!-- optional -->

## Considered Alternatives

* *[alternative 1]*
* *[...]* <!-- numbers of alternatives can vary -->

## Decision Outcome

* Chosen Alternative: *[alternative 1]*
* *[justification. e.g., only alternative, which meets k.o. criterion decision
    driver | which resolves force force | ... | comes out best (see below)]*
* *[consequences. e.g., negative impact on quality attribute,
    follow-up decisions required, ...]* <!-- optional -->

## Pros and Cons of the Alternatives <!-- optional -->

### *[alternative 1]*

* `+` *[argument 1 pro]*
* `-` *[argument 1 con]*
* *[...]* <!-- numbers of pros and cons can vary -->
```

Fig. 2. MADR 1.0.0 format decomposing the Y-Statement elements into document sections.

(e.g., `adr -l "1:Amends:Amended by" Use PosgreSQL`, tells `adr-tools` that ADR-0001 is amended by the newly created ADR).

An index of existing architectural decision records is a welcome feature to gain an overview of the decision making status and be able to navigate the log efficiently. The existing `adr-tools` already offers the command `adr generate toc`. This, however, generates a completely new file and does not allow to add arbitrary text before or after the toc. For the generation of the table of contents of one markdown file, the tool `markdown-toc`[8] inserts the TOC after the token `<!-- toc -->`. Inspired by that idea, we implemented `adr-log`, which places the list of all ADRs after the placeholder `<!-- adrlog -->`. We chose the name "log" instead of "toc" to be consistent with the database terminology, where a set of records forms a log.

[8] `https://www.npmjs.com/package/markdown-toc`

5 Preliminary Validation

We validated MADR and tooling in action research [3] on the Eclipse Winery project that is driven by one architect, three coding architects, and two developers (students, staff members, and volunteers).

In action research, the researcher joins a project and influences it actively, for instance as coach, pacemaker, or technical reviewer[9]. Applying action research allowed us to experience the practical applicability of our concepts ourselves and to interact with and learn from other users while they used MADR. We followed a very basic study protocol of 1) define validation goals and approach, 2a) present MADR and `adr-tools` to the project team and create a first MADR record ourselves (lead by example), 2b) monitor usage and remind project participants, e.g. in sprint planning meetings and retrospectives, 2c) give feedback to project participants and offer coaching, and finally 3) collect data and seek suggestions for improvements from project participants.

A total of 16 MADR records were created.[10] Eight of these fully filled out the template, and eight used a shortened form without the explicit section "Pros and Cons of the Alternatives". Two of the short forms additionally include details of the solution. When working with code, it was easy to document the decision along with the code. A single file had to be copied and renamed (or `adr new TITLE` invoked). Then, one could start with writing down the context, options, chosen option, and the pros and cons.

Seasoned professionals did not have issues to fill out the template and even came up with their own. Inexperienced students were able to document their decisions ranging from supported writing to independent writing. A major issue for them was to understand how to replace the placeholders in markdown. For instance, in ADR-0005[11], the options listed there did neither have ids nor short titles. The chosen option was referred to as "Option D", but there was no explicit option D — only a fourth unnamed option. Some students also reported that they were afraid to be criticized for options not considered. Since MADR makes it explicit which options a solution was chosen from, it is easy to detect if an important option was missed. On the positive side, this leads to a teaching effect and allows supervisors to get to know which knowledge the students lack at a certain educational level. On the one hand, it was agreed, that is difficult to create an ADR if the technology itself is new. On the other hand, it was also agreed that is necessary to document decisions after one has enough knowledge (e.g., after experimenting longer with different options) to make it feasible for others to understand the decisions taken. This is in line with Parnas's view on a rational software process [11].

In summary, the users reported that the MADR template and tools helped them to be clear about the available options and to choose the best fitting one based on arguments. The template was filled during the discussions and helped to refine the pros and cons of alternatives.

---

[9] This is different from exposing selected research results to users and merely observing them (this would be done in a controlled experiment).

[10] `https://github.com/eclipse/winery/tree/d84b9d7b6c9828fd20bc6b1e2fcc0cf3653c3d43/docs/adr`

[11] `https://github.com/eclipse/winery/blob/d84b9d7b6c9828fd20bc6b1e2fcc0cf3653c3d43/docs/adr/0005-XML-editor-does-not-enforce-validation.md`

6 Discussion

Feedback from reviews and workshop raised some concerns whether placing ADRs in a single folder really scales: a complex system may consist of multiple microservices, and each microservice can itself be structured in different modules even written in different languages. Thus, the granularity of the decisions is different. Two possible solutions are: A) adding a category to each ADR and offer filtering. B) putting each ADR close to the source code where the decision is taken, e.g., `src/doc/adr` for a Java project.

A developer began to add longer explanations of code howtos to the ADR. The reasoning was that this code howto is very related to the ADR and that there is one place where the decision and the coding consequences can be found. Thus, an interesting question requiring further investigation and discussion would be whether close-to-code (M)ADR documentation leads to an increased use of documentation (in comparison to external documentation).

The presented version 1.0.0 of MADR uses slightly different terms than the Y-Statements (Sect. 3). We plan to refactor future versions of MADR[12] to be even closer to the terms of Y-Statements as these are proven in industry projects and have been gaining momentum recently [6].

In large projects, it is common to create a project management issue for each change. In MADR, the link to an issue is optional to enable application in small projects. These two different settings call for MADR *profiles*. For instance, one such profile could enforce the link to the ticket/issue number (pointing to an entry in task management system) and make the section "pros and cons of the alternatives" mandatory.

7 Conclusion and Outlook

This paper presented Markdown Architectural Decision (MADR) records, a decision capturing template derived from earlier work on a Y-statement format. We also presented an extension of existing `adr-tools` to enable command-line tools for handling MADRs as well as a new `adr-log` tool to generate a list of existing ADRs.

Based on the early feedback, we plan to improve the creation and review process. We also consider to develop a comprehensive yet lean getting started tutorial and quick reference card.

MADRs capture a concrete decision in the context of a single particular project. However, problems and options may reoccur and different options might be chosen in different contexts. For instance when a system runs normally in the absence of partitions, one choose between different trade offs between latency and consistency [1]. Each trade off has its pros and cons which are differently weighted in each context. Currently, it is possible to model this "problem space" using AD-Mentor [18], but not using Markdown. To come up with a corresponding Markdown format and tool integration for knowledge sharing and reuse therefore is an enhancement to be considered in the future evolution of MADR.

[12] The development of MADR takes place at `https://github.com/adr/madr/`.

Acknowledgments This work is partially funded by the BMWi projects SmartOrchestra (01MD16001F) and IC4F (01MA17008G).

References

1. Abadi, D.: Consistency Tradeoffs in Modern Distributed Database System Design: CAP is Only Part of the Story. Computer 45(2), 37–42 (2012)
2. Alexeeva, Z., et al.: Design Decision Documentation: A Literature Overview. In: Software Architecture, pp. 84–101. Springer International Publishing (2016)
3. Avison, D., et al.: Action Research. Communications of the ACM 42(1), 94–97 (1999)
4. Capilla, R., Jansen, A., Tang, A., Avgeriou, P., Babar, M.A.: 10 Years of Software Architecture Knowledge Management: Practice and Future. Journal of Systems and Software 116, 191–205 (jun 2016)
5. Fowler, M., Rice, D.: Patterns of Enterprise Application Architecture. Addison-Wesley, Boston, Mass. (2003)
6. Harrer, S., Lenhard, J., Dietz, L.: Java by Comparison: Become a Java Craftsman in 80 Examples. Pragmatic Bookshelf (2018), http://java.by-comparison.com
7. ISO/IEC/IEEE 42010:2011: Systems and software engineering – Architecture description. Standard (Dec 2011)
8. Leschner, J.: Simpler GitHub Pages publishing (2016), https://github.com/blog/2228-simpler-github-pages-publishing
9. MacLean, A., Young, R.M., Bellotti, V.M.E., Moran, T.P.: Questions, Options, and Criteria: Elements of Design Space Analysis. Hum.-Comput. Interact. 6(3), 201–250 (Sep 1991)
10. Nygard, M.: Documenting architecture decisions (2011), http://thinkrelevance.com/blog/2011/11/15/documenting-architecture-decisions
11. Parnas, D.L., Clements, P.C.: A rational design process: How and why to fake it. In: Formal Methods and Software Development, pp. 80–100. Springer Science + Business Media (1985)
12. Schubanz, M.: Full Decision Capture Template (2017), https://github.com/schubmat/DecisionCapture/blob/ca03429634ac2779b37e12aee34dd09a5fdbdcd3/templates/captureTemplate_full.md
13. ThoughtWorks: Technology Radar Vol. 17, https://thoughtworks.com/radar
14. Van Heesch, U., et al.: Software Specification and Documentation in Continuous Software Development: A Focus Group Report. In: 22nd European Conference on Pattern Languages of Programs (EuroPLoP'17). ACM (2017)
15. Zdun, U., et al.: Sustainable Architectural Design Decisions. IEEE Software 30(6), 46–53 (2013)
16. Zimmermann, O.: Making Architectural Knowledge Sustainable – Industrial Practice Report and Outlook (2012), presentation at SATURN 2012, http://www.sei.cmu.edu/library/abstracts/presentations/zimmermann-saturn2012.cfm
17. Zimmermann, O., Koehler, J., Leymann, F.: Architectural Decision Models as Micro-Methodology for Service-Oriented Analysis and Design. In: Workshop on Software Engineering Methods for Service Oriented Architecture 2007 (SEMSOA). CEUR (2007)
18. Zimmermann, O., et al.: Architectural Decision Guidance Across Projects – Problem Space Modeling, Decision Backlog Management and Cloud Computing Knowledge. In: 12th Working IEEE/IFIP Conference on Software Architecture (WICSA). IEEE (2015)

All links were last followed on February 14, 2018.

CloudRef – Towards Collaborative Reference Management in the Cloud

Oliver Kopp[1], Uwe Breitenbücher[2], and Tamara Müller[2]

[1] IPVS, University of Stuttgart, Germany, kopp@ipvs.uni-stuttgart.de
[2] IAAS, University of Stuttgart, Germany, {lastname}@iaas.uni-stuttgart.de

Abstract. With the help of literature management software, references can be collected, managed, and exported in bibliographies. Online resources offer functionalities to import references into reference management tools. However, the entries are often incomplete or faulty. CloudRef proposes to solve this issue by employing votings for new references and updates of references. To further foster collaboration, comments on PDFs can be shared among the users of CloudRef.

1 Introduction

When writing a scientific paper one has always to deal with an plethora of literature on the topic. Reference management software was invented to support researchers in that regard: The tools are used to collect literature, manage references, and export bibliographies. They provide an efficient way to keep an overview of a large amount of literature. Numerous tools provide the opportunity to manage knowledge about references inside comments, notes, or tags. Researching is often a collaborative task demanding that literature management software should support collaboration. This includes sharing references and comments with other users or people who use another literature management software. Sharing comments with others may be beneficial 1) to ease understanding the paper itself and 2) to ease finding relevant papers, because indexable text is provided.

There are multiple resources on the web offering searching for literature such as "Google Scholar"[3] or "The Collection of Computer Science Bibliographies"[4]. Many of them offer the functionality to import a reference into the preferred reference management software. However, they often provide incomplete or faulty reference entries. One exception is MathSciNet, where more than 20 persons take care of the quality of the references [8]. This quality assurance, however, is not implemented by all publishers.

A correct and complete entry is required for a correct reference list, which is a prerequisite for high-quality publications. Many programs for managing references provide a mechanism to detect missing required fields and highlight these entries to show the user that they are incomplete. However, this is not sufficient, because wrong information is not detected and it is tedious to find the correct missing information. The users have to check each reference entry manually to ensure correctness.

[3] https://scholar.google.com
[4] https://liinwww.ira.uka.de/bibliography

N. Herzberg, C. Hochreiner, O. Kopp, J. Lenhard (Eds.): 10th ZEUS Workshop, ZEUS 2018, Dresden, Germany, 8-9 February 2018, published at http://ceur-ws.org/Vol-2072

Hence, the goal of CloudRef is to provide a cloud-based web application for collaborative reference management with to main features:

1. CloudRef provides quality assurance by voting on bibliographical references to ensure complete and faultless references.
2. To support the cooperation of several people CloudRef enables to post comments to literature at different levels of visibility.

Sect. 2 presents related work on the field. Subsequently, Sect. 3 outlines the demonstration of CloudRef followed by a description of the implementation (Sect. 4). Finally, Sect. 5 provides a discussion and an outlook on future work.

2 Related Work

SoJa (Social JabRef [7]) introduces a social network among users. To establish a source of high-quality entries, for each topic, a user maintaining these entries has to be chosen in the community. There is no voting mechanism in place and comments can only be shared by embedding them into the BibTeX file. Haase et al. [10] assume that there are BibTeX databases with high-quality entries and that the issue is to identify duplicates and to find entries. To tackle these issues, they present the system Bibster. SharRef [22] focuses on sharing bibliographic data among groups and offers both a Web-based Client and a Java-based Desktop Client. Quality of entries is assured by having bibliography entries and automatically-updated shadows of them. There is no internal voting or commenting system in place. SocioBiblog [17] relies on the authors publishing their bibliographic data correctly on their homepages. There is no way presented on how to ensure quality of the resulting BibTeX entries. BibSonomy [3] and its variant PUMA [4] offer to collect publications. It is possible to edit bibliographic data [6]. While there is a history function[5], all edits are immediatly visible so there is no suggestion process as we propose. Academic search engines are surveyed by Ortega [16]. Additionally, there is OverCite [19] aggregating search results in a CiteSeer-like way. These tools offer search capabilities only and not a defined way to correct bibliographic entries. Beraka et al. [5] present a system for exchanging bibliographic information of scientific review and survey articles. Users can approve or disapprove bibliographic entries, but it is unclear how contradicting votes are treated. For presentation of surveys, SurVis [1] can be used. It is a read-only system without built-in functionalities of ensuring high-quality bibliographic data. Tkaczyk et al. [20] surveyed on reference parsers. They convert free reference text to a structured format. Thus, this is a way to get bibliographic data into a literature management system, but it is not ensured that the parsed data is of high quality itself. There is a movement on correctly citing software [18]. However, there is no quality control process proposed. Finally, we investigated 15 popular literature management systems[6] and none of them offers 1) a voting system on bibliography entries and 2) comments with dedicated visibility.

[5] Example: `https://www.bibsonomy.org/history/bibtex/57fe43734b18909a24bf5bf6608d2a09`

[6] `https://ultimate-comparisons.github.io/ultimate-reference-management-software-comparison`

3 Demonstration

After startup, the current prototype CloudRef shows a table of all references (Fig. 1). References with a green check mark have been reviewed and marked as high-quality. A new reference can be input using a form-based editor or by uploading a BibTeX file. A

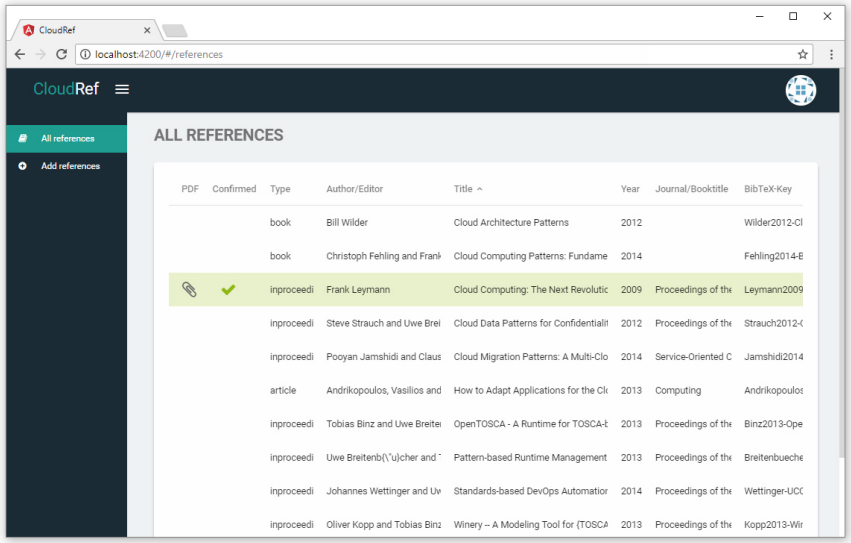

Fig. 1. Entry table showing references.

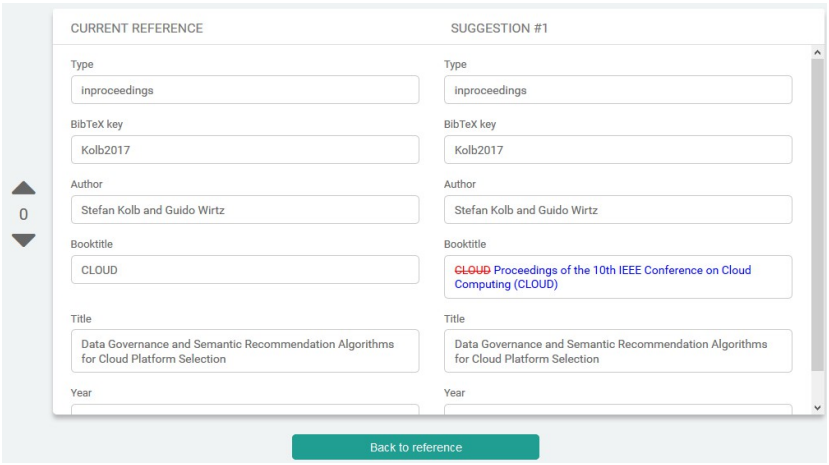

Fig. 2. Suggestion for modification with voting possibility. The dialog is based on JabRef's Merge Entries Dialog, `https://help.jabref.org/en/MergeEntries`.

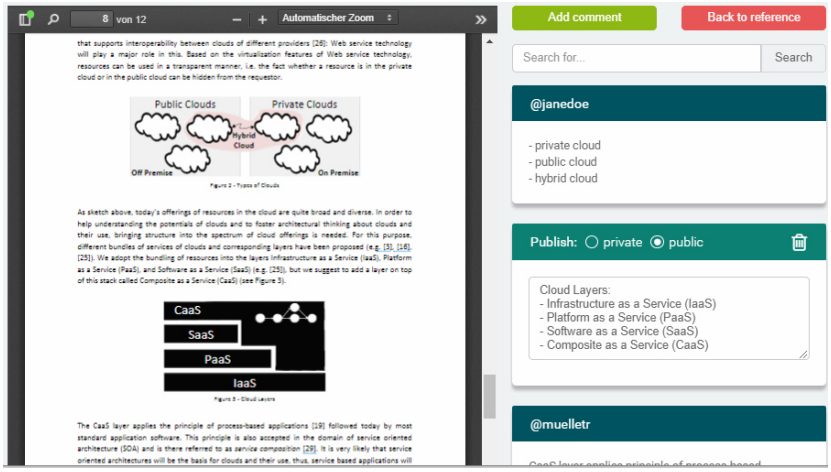

Fig. 3. PDF comments.

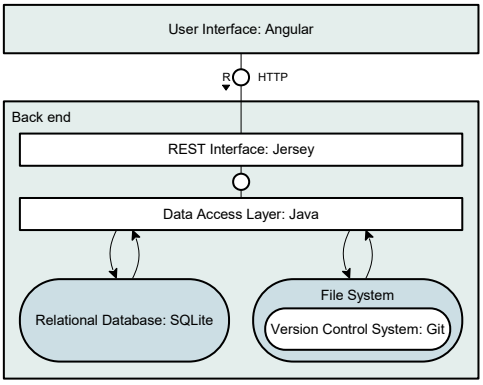

Fig. 4. Architecture of the CloudRef platform with implementation details. Notation: FMC [12].

suggestion for improvements can be done using the entry editor. A user can click on "See suggestions for modifications" and a respective dialog is shown (Fig. 2). On the left side, there are the voting buttons. The number indicates the number of votes. After five positive votes by different users have been reached, the suggestion is merged.

In case a PDF is attached, comments on the PDF can be made (Fig. 3). A user can set the visibility of his own comments to public or private. "Public" denotes that each logged in user can see the comment and "private" denotes that the user exclusively can see the comment. This helps newcomers to research to make private notes and the more experienced researches to share their comments.

4 Implementation

The architecture of CloudRef including the used technologies is presented in Fig. 4. We implemented CloudRef using Java and Angular. Regarding the storage, we decided to

put the comments into a SQL database, placing the PDFs directly into the filesystem, and versioning the bibliography entries using git. Each suggestion becomes a new branch in git. In case a suggestion is accepted, the branch is merged. This way, we did not have to reinvent the whole branching and merging concept, but could rely on git's possibilities. The implementation is published at `https://github.com/JabRef/cloudref/` and is offered as Docker image at `https://hub.docker.com/r/jabref/cloudref/`.

5 Discussion and Outlook

Currently, CloudRef is an initial idea for collaborative reference management. CloudRef is currently targeted at research groups with around ten persons. This ensures that quality standards established in a group is followed. The group, however, has to define its quality standards. For instance, there is currently no global agreement whether one should always put the abbreviation of a conference name in parentheses to the end of the conference title. The next natural step is to create a style guide for BibTeX and to integrate a checker into CloudRef similar to a GitHub pull request status checker [21].

When moving to a larger user base or offering CloudRef as public SaaS offering, additional concepts for reviewing and maintaining references have to be developed and the current voting concept has to be evaluated. For instance, the number of required positive votes is a variable to evaluate.

The history of each BibTeX entry is stored in the backend. Since there might be different views on a BibTeX entry, CloudRef should be able to show the history to the user. This enables him to propose another version of the entry based on historic proposals.

To provide more features for high-quality references, we plan to use the logic package of JabRef[7], transpile it to JavaScript using jsweet[8], and embed it in CloudRef. We also plan to integrate CloudRef's functionality into JabRef.

CloudRef is currently targeted as "Multiple Instances Service" [15]. To enable it being hosted using as "Arbitrary Instances Service," we are going to work on the backend implementation and make CloudRef a real cloud-native application [14]. This especially includes exchanging the storage layer by a PaaS one [13].

Finding related work is still a challenging task. To ease this, we aim for integrating a) the recommender system Mr. DLib [2] into CloudRef in a similar way it has been done for JabRef [9] or b) the user profile recommendations implemented by Bibster [11].

References

1. Beck, F., Koch, S., Weiskopf, D.: Visual analysis and dissemination of scientific literature collections with SurVis. IEEE Transactions on Visualization and Computer Graphics 22(1), 180–189 (Jan 2016)
2. Beel, J., Gipp, B., Langer, S., Genzmehr, M., Wilde, E., Nürnberger, A., Pitman, J.: Introducing Mr. DLib, a Machine-readable Digital Library. In: 11th annual international ACM/IEEE joint conference on Digital libraries (JCDL'11). ACM (2011)

[7] `https://www.jabref.org`

[8] `http://www.jsweet.org`

3. Benz, D., Hotho, A., Jäschke, R., Krause, B., Mitzlaff, F., Schmitz, C., Stumme, G.: The social bookmark and publication management system bibsonomy. The VLDB Journal 19(6), 849–875 (Dec 2010)
4. Benz, D., Hotho, A., Jäschke, R., Stumme, G., Halle, A., Lima, A.G.S., Steenweg, H., Stefani, S.: Academic Publication Management with PUMA – Collect, Organize and Share Publications. In: Research and Advanced Technology for Digital Libraries, pp. 417–420. Springer (2010)
5. Beraka, M., Al-Dhelaan, A., Al-Rodhaan, M.: A Bibliographic System for Review/Survey Articles. In: 2014 International Conference on Information Science & Applications (ICISA'14). IEEE (2014)
6. BibSonomy Team: Community posts (2018), https://www.bibsonomy.org/help_en/CommunityPosts
7. Datta, A.: SoJa: Collaborative reference management using a decentralized social information system. In: 6th International ICST Conference on Collaborative Computing: Networking, Applications, Worksharing (CollaborateCom 2010). IEEE (2010)
8. Dunne, E.: References and Citations (2015), https://blogs.ams.org/beyondreviews/2015/07/11/references-and-citations/
9. Feyer, S., Siebert, S., Gipp, B., Aizawa, A., Beel, J.: Integration of the Scientific Recommender System Mr. DLib into the Reference Manager JabRef. In: 39th European Conference on IR Research (ECIR 2017). Springer (2017)
10. Haase, P., Broekstra, J., Ehrig, M., Menken, M., Mika, P., Olko, M., Plechawski, M., Pyszlak, P., Schnizler, B., Siebes, R., Staab, S., Tempich, C.: Bibster – A Semantics-Based Bibliographic Peer-to-Peer System. In: 3rd International Semantic Web Conference (ISWC 2004). Springer (2004)
11. Haase, P., Stojanovic, N., Voelker, J., Sure, Y.: Personalized information retrieval in bibster, a semantics-based bibliographic peer-to-peer system. In: 5th International Conference on Knowledge Management (I-KNOW'05). p. 104 (2005)
12. Keller, F., Wendt, S.: FMC: an approach towards architecture-centric system development. In: 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems. IEEE (2003)
13. Kolb, S., Lenhard, J., Wirtz, G.: Application Migration Effort in the Cloud. Services Transactions on Cloud Computing (STCC) 3(4), 1–15 (2015)
14. Leymann, F., Wettinger, J., Wagner, S., Fehling, C.: Native Cloud Applications: Why Virtual Machines, Images and Containers Miss the Point! In: 6th International Conference on Cloud Computing and Services Science. SCITEPRESS (2016)
15. Mietzner, R., Leymann, F., Unger, T.: Horizontal and vertical combination of multi-tenancy patterns in service-oriented applications. Enterprise Information Systems 5(1), 59–77 (Feb 2011)
16. Ortega, J.L.: Academic Search Engines: A Quantitative Outlook. Chandos Publishing (2014)
17. Shakya, A., Takeda, H., Wuwongse, V., Ohmukai, I.: SocioBiblog: A decentralized platform for sharing bibliographic information. In: IADIS International Conference WWW/Internet (2007)
18. Smith, A.M., Katz, D.S., and, K.E.N.: Software citation principles. PeerJ Computer Science 2, e86 (sep 2016)
19. Stribling, J., Li, J., Councill, I.G., Kaashoek, M.F., Morris, R.: Overcite: A distributed, cooperative citeseer. In: 3rd Symposium on Networked Systems, Design & Implemention (NDSI'06) (2006)
20. Tkaczyk, D., Collins, A., Sheridan, P., Beel, J.: Evaluation and Comparison of Open Source Bibliographic Reference Parsers: A Business Use Case (2018), arXiv:1802.01168v1
21. Vasilescu, B., Yu, Y., Wang, H., Devanbu, P., Filkov, V.: Quality and productivity outcomes relating to continuous integration in GitHub. In: 10th Joint Meeting on Foundations of Software Engineering (ESEC/FSE 2015). ACM (2015)
22. Wilde, E.: Personalization of Shared Data: The ShaRef Approach. Journal of Digital Information 8(3) (2007)

Author Index