Extraction of Configurable and Reusable Microservices from Legacy Systems: An Exploratory Study

Luiz Carvalho
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Rio de Janeiro, Brazil
lmcarvalho@inf.puc-rio.br

Alessandro Garcia
Pontifical Catholic University of Rio de Janeiro
Rio de Janeiro, Rio de Janeiro, Brazil
afgarcia@inf.puc-rio.br

Wesley K. G. Assunção
Federal University of Technology - Paraná
Toledo, Paraná, Brazil
wesleyk@utfpr.edu.br

Rodrigo Bonifácio
University of Brasília
Brasília, DF, Brazil
rbonifacio@unb.br

Leonardo P. Tizzei
IBM Research
São Paulo, São Paulo, Brazil
ltizzei@br.ibm.com

Thelma Elita Colanzi
State University of Maringá
Maringá, Paraná, Brazil
thelma@din.uem.br

ABSTRACT

Microservices is an emerging industrial technique to promote better modularization and management of small and autonomous services. Microservice architecture is widely used to overcome the limitations of monolithic legacy systems, such as limited maintainability and reusability. Migration to a microservice architecture is increasingly becoming the focus of academic research. However, there is little knowledge on how microservices are extracted from legacy systems in practice. Among these limitations, there is a lack of understanding if variability is considered useful along the microservice extraction from a configurable system. In order to address this gap, we performed an exploratory study composed of two phases. Firstly, we conducted an online survey with 26 specialists that contributed to the migration of existing systems to a microservice architecture. Secondly, we performed individual interviews with seven survey participants. A subset of the participants (13 out of 26) dealt with systems with variability during the extraction, which stated that variability is a key criterion for structuring the microservices. Moreover, variability in the legacy system is usually implemented with simple mechanisms. Finally, initial evidence points out that microservices extraction can increase software customization.

CCS CONCEPTS

• **Software and its engineering** → **Software architectures**; **Maintaining software**; **Software product lines**; *Reusability*; Software evolution.

KEYWORDS

Microservice architecture, microservice customization, software variability, architecture migration

SPLC '19, September 9–13, 2019, Paris, France

ACM ISBN 978-1-4503-7138-4/19/09…$15.00
https://doi.org/10.1145/3336294.3336319

ACM Reference Format:
Luiz Carvalho, Alessandro Garcia, Wesley K. G. Assunção, Rodrigo Bonifácio, Leonardo P. Tizzei, and Thelma Elita Colanzi. 2019. Extraction of Configurable and Reusable Microservices from Legacy Systems: An Exploratory Study. In *23rd International Systems and Software Product Line Conference - Volume A (SPLC '19), September 9–13, 2019, Paris, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3336294.3336319

1 INTRODUCTION

A legacy system, commonly found in industry, represents a long term massive investment. Despite of their business importance, legacy systems are difficult to extend and include innovation [5]. In addition, these systems usually have a monolithic architecture, with components tangled in a single unit, strongly connected and interdependent [20, 27]. Currently, many companies have been adopting microservice architectures to modernize monolithic legacy systems [6, 14, 17, 20, 28, 30]. Microservices are small and autonomous services that work together [23]. The benefits of adopting microservices are: reduced effort for maintenance and evolution, increased availability of services, ease of innovation, continuous delivery, ease of DevOps incorporation, facilitated scalability of parts with more demand, etc [20, 27].

Service-based architectures must meet some attributes to satisfactorily fulfill their purpose [10]: (i) scalability, enabling optimization in the use of hardware resources to meet high demand services; (ii) multi-tenant efficiency, offering transparency in the use of shared services, since the service should optimize the sharing of resources and also isolate the behavior of different tenant; and (iii) variability, where a single code base provides common and variable functionalities to all tenants. Microservices are known to work well with the first two attributes [28]. However, the literature is scarce in relation to the use of variability. A possible explanation for the lack of discussion may be the fact that this technology was initially conceived in the industry and only in the last years have had attention of academia[1].

Configurable systems are widely developed in the industry to meet demands related to different types of hardware, platforms, serving diverse customers or market segments, etc [29]. In this context, we need to understand how functionalities should be modularized and customized as microservices to fulfill customer needs.

[1] https://martinfowler.com/articles/microservices.html

The goal of this paper is to investigate the importance of variability on the process of extracting microservices from monolithic legacy systems in industry. To this end, we analyzed 26 survey responses and seven interviews with practitioners. Among the 26 participants, 13 dealt with variability during the extraction and they stated that variability was a key criterion for structuring the microservices. Moreover, we observed an increase of requests for customization after microservices extraction that initially were not configurable. In this way, initial evidence points out that the microservices extraction can increase software customization.

This paper is organized as follows. Section 2 presents the background. Section 3 details our study. Results and discussion are in Section 4. Sections 5 and 6 address threats to validity and related work, respectively. Section 7 concludes the paper.

2 BACKGROUND

This section presents the main concepts involved in this paper, such as microservices, customization, and variability.

2.1 Microservices

A microservice is expected to have fine granularity [11] and be autonomous: (i) it should consist of a service highly independent from others, and (ii) it should enable an independent choice of technologies. For communication, microservices architecture commonly adopted lightweight protocols.

Not rarely, microservices are not developed from scratch, but they result from the migration of existing systems [14, 20, 23]. However, the migration from an existing system to microservices is perceived as challenging by developers who have experienced it [20, 27]. Previous studies indicate that developers use the source code of the existing system in migration to microservice architecture [15]. In the same way, approaches were proposed using criteria observed in source code [18, 21], generally using coupling and cohesion criteria. Others approaches recommend observing the database schema [13, 23]. However, there is a lack of the comprehension of variability usefulness in the migration process when the existing system is a configurable system or software product line.

2.2 Customization and Variability

Service-oriented architectures must allow tenant-specific configuration and customization. The tenant-specific adaptations may affect all layers of the application, from functional requirements to database schema. Furthermore, tenants do not only have different requirements regarding functional properties, they can also require different non-functional requirements of service properties [22].

To achieve such customization, industry must deal with software variability, which is the ability of a system or an asset to be adapted for using in a particular context [3]. Variability can be viewed as consisting of two dimensions [8]. The space dimension is concerned with the use of software in multiple contexts. The time dimension is concerned with the ability of software to support evolution and changing requirements in its various contexts.

3 EXPLORATORY STUDY DESIGN

The goal of our exploratory study is to better understand the migration process to microservice architecture and how software variability is useful in the decision-making process. In what follows, we show in Subsection 3.1 the research questions and their motivations. Subsection 3.2 presents the two phases (survey and interview), population, and sampling. In addition, Subsection 3.3 shows the instruments used in the survey and interview.

3.1 Research Questions

Based on the aforementioned goal, we intend to answer the following research questions:

RQ1 How important is variability in the migration process to the microservice architecture?

- **RQ1.1** Did the original systems contain some sort of customization prior to the migration?
- **RQ1.2** What mechanisms were used to implement variability prior to the migration to microservice architecture?
- **RQ1.3** Do developers consider variability as useful criteria in the migration process to microservice architecture?

RQ2 How is variability present after migration to the microservice architecture?

- **RQ2.1** Does microservices extraction increase software customization?
- **RQ2.2** What mechanisms are used to software customization in microservice architecture?

In summary, for RQ1 we want to investigate the relevance of variability and its mechanisms to support and guide the migration of configurable systems to a microservice architecture. RQ2 aims to understand how variability is present after the migration to a microservice architecture, since there is a lack of understanding about how configurable systems coexist with the microservice architecture.

3.2 Study Phases, Population and Sample

Our exploratory study is composed of two phases: (i) a survey with specialists, and (ii) interviews with specialists who answered the survey. The target of our survey are practitioners and industrial researchers with a background in migrating existing systems to microservices. We used the results of three mapping studies [1, 16, 24] to identify an initial target population. In addition, we performed a snowballing search [31] in studies that cited the referred mapping studies. For performing this search, we used Google Scholar[2].

Our strategy to contact specialists was to invite the authors of the papers found in the three mapping studies to participate in the survey. We requested that they submit the survey to the subjects of the empirical studies. After execution, our search plan resulted in the recruitment of 90 subjects. The survey was executed from January to March 2019. From the subjects recruited, we collected answers from 26 participants, resulting in a participation rate of 29%. Of which 13 participants also answered our questions related to software variability. To carry out deep analysis, we invited the survey participants to an interview. We conducted 7 interviews.

[2]https://scholar.google.com/

3.3 Instrumentation

Survey. We organized the survey into two groups. The first group is composed of questions for characterizing the participants. We asked the academic background, development experience, and position in the current job. We also inquired about their background in migrating existing systems to microservices.

The second group of questions has the objective of perceiving the usefulness of variability, in the extraction of microservices. For this group of questions, we used a five-point Likert scale associated with the levels of usefulness for variability, from the least (1) to the most (5). We also asked the participants to justify their answers. To avoid misinterpretations, we provided in the questionnaire a definition of variability, which is the ability to derive different products from a common set of artifacts [2].

Interview. The goal of this phase was to understand the post characteristics of the migration process. The interviews were conducted using video conference tools. Each interview was conducted by an interviewer and a scribe that took notes. For the execution of the interview, we chose a semi-structured approach. That is, questions that answers can be quantified (structured) and suggest the theme (unstructured) [26].

Regarding the questions, the participants were first questioned with open and general questions about more high-level themes, in an unstructured way. For example, explaining about the existing system that was/is being migrated to microservice architecture. After that, we inquired them with quantitative questions. For example, the number of microservices extracted. This approach was chosen by observing that survey participants had an experience mean of 15.77 years in software development. Moreover, participants have been involved in the migration process to the microservice architecture. In the questions, we investigated the themes: (i) existing systems, (ii) migration process, (iii) variability, and (iv) tools.

4 RESULTS AND ANALYSIS

In this section, we present the results and their analysis. The first phase of our study (survey) offers a better understanding about the usefulness of variability.

In a previous study, we observed how relevant are seven criteria during migration to microservices architecture. However, the previous study did not perform an analysis of variability [9]. This happened because of the low number of participants with expertise on migrating some existing system with variability to a microservice architecture. In this present work, the greater number of survey answers allowed us made analyses about variability in the first phase. Moreover, the second phase of study (interview) provided initial results about the post-migration process to microservice architecture. Among them, the request for increased customization of the microservices in order to deal with different customers or groups, in which, before the migration process was not made.

4.1 Participants Characterization

Survey respondents experience may be evidenced by previous and ongoing activities with the migration process to microservice architecture. The vast majority of the respondents had already concluded their participation in at least two migration process (mean of 2.5 and median of 2.0) and most of them are actively participating in at least one project (mean of 1.3 and median of 1.0) where the microservices extraction is currently underway. Besides that, participants observed at least one process (a mean of 1.5 and median of 1.0) and they are observing at least one migration (a mean of 1.5 and median of 1.0). The respondents have extensive experience in software development. The time in years the participants have been developing software is considerably high: (i) the mean is 15.77 years and median is 15 years, and (ii) with a maximum of 35 years and a minimum of 3 years. Our sample of respondents is diversified in terms of the roles that they play in their employment: developers (42%), architects or engineers (23%), team leaders (19%), and industry researchers (19%).

4.2 Variability

From the 26 participants, 50% have never considered variability. For these respondents, the domain they work does not require the ability to derive different products from a common set of artifacts [2]. Their intention was to specifically migrate a single software system to a microservice architecture. However, the other half of the participants have answered that their existing systems, which were the target of microservices extraction, had variability. Thus, these respondents were able to answer about the usefulness of variability during the migration process to a microservice architecture.

Among the participants with previous experience to answer on the usefulness of this criterion (13 out of 26) (see Figure 1): (i) 31% considered it as not relevant by assigning values of 1 or 2 in the Likert scale, and (ii) 69% of the participants that migrated systems with variability considered it as useful or very useful. There is no response with moderate usefulness. Among those that considered variability important, one participant said: *"It was useful to identify the variabilities and features of each product"*. Other participant said that *"There are 3 systems that will be affected by the migration and we need to ensure the differences between them and how these differences can be handled by the migration process"*. In this way, most of the participants that migrated systems with variability consider variability useful or very useful in the migration process to the microservices architecture. However, previous approaches to microservices extraction do not make use of variability criterion.

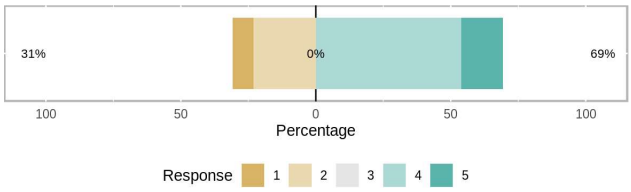

Figure 1: The usefulness of variability for the migration process to microservices architecture

Regarding the variability implementation approaches, the most common ones are those shown in Table 1. In the last column of the table, we present the proportion in which they were cited.

Version control was the most common variability implementation approach used in existing systems, that is, in systems before the migration to microservice architecture. One might consider this is the most simplistic approach as it leads to the management

Table 1: Variability Implementation Approaches

Approach	Percentage
Version control	77%
Design patterns	62%
Framework	62%
Components	54%
Parameters	46%
Build systems	38%
Aspect-oriented programming	31%
Feature-oriented programming	15%
Pre-processor	8%

of copy and paste of different products. For example, without the need of extending the programming language being adopted as in feature-oriented programming. Approaches more sophisticated are less common ones, like the use of aspect-oriented programming and feature-oriented programming. The mean of variability implementation approaches used is 4.5. All the respondents mentioned more than one approach to implement variability.

The survey results enable us to answer **RQ1** as follows:

> ***RQ1.1:*** *50% of the participants answered that at least one of their existing systems (from which microservices were extracted) had variability.*
> ***RQ1.2:*** *Version control is the most used mechanism (77%) to implement variability in those existing systems.*
> ***RQ1.3:*** *69% of the participants that migrated systems with variability considered that variability is a useful or very useful criterion during the migration process.*

4.3 Microservice Customization

During the interview phase, we inquired the participants whether the extraction allowed that some microservices could be used in different contexts. To our surprise, four (57%) of the interviewed participants said that some customization was required after this migration process to microservice architecture. The customization was needed to attend requests of different groups inside the same enterprise or customers of the software system. The four cases previously answered in the survey that the systems prior to migration to microservices had no variability. This seems to be an indication that the process of migration to microservices leverages the customization of the single system by increasing modularity and more interfaces available to its customers. Common ways to manage these new requirements for dealing with customization of the software systems are reported in the following paragraphs.

Regarding the other three participants (43%) of the interview, two of them reported in both the survey and the interview that: (i) the system has variability before the migration to microservice architecture, (ii) and the system continued to have variability after the migration. In one of these cases, the variability in the migrated system relied on design patterns and parameters in the interface to identify a tenant [22]. In the other case, the migration process is still underway. However, it is expected that some microservices will not exist or have their behavior drastically modified in different usage contexts. In the third case, which there was no variability before or after the migration, the software system was completely developed from scratch; then, the system was migrated afterward to satisfy a single customer. It was not used by different groups of the enterprise.

> ***RQ2.1:*** *Initial evidence points out that the migration process to a microservices architecture increases the customizations of the system.*

During the interviews, four participants reported that the migration to microservices was not made with the goal of turning the system more easily customizable. However, after the migration, customization requests emerged by different clients or groups within the same enterprise.

Among the four cases that reported this growth for post-migration customization, three of them described which approaches were adopted to implement variability in a microservice architecture. One of the participants reported that the need for customization emerged after the microservices extraction; however, he did not observe or participate in the process of implementing or managing the customizations.

We describe below the three implementation approaches employed to deal with the customization required after the microservices extraction: *Copy and Paste*, *Big Interface*, and *Filtering to the Different Platforms*.

Copy and Paste was adopted after the migration process to microservice architecture by one of the participants. That happened when customizations in the interface of extracted microservices began to be requested by different groups within the same enterprise. In this case, it was decided to copy the microservices implementation and perform the customization. Thus, the copied microservice coexists with the original one (inclusive at runtime). Changes are performed in both microservices by the original developers when maintenance is required in mandatory features. It was also reported that code of test cases is submitted to the same copy and paste process and it is managed in the same way. This approach is often used in the industry as it promptly promotes reuse of verified functionalities, without requiring high upfront investment and achieving short-term benefits, as reported by Dubinsky *et al.* [12].

Big Interface was also an adopted solution. In this case, certain consumers of the microservice APIs needed additional information, or the same information in different formats. To fulfill this demand, the developers just included all required information in already existing APIs. However, it should be noted that this solution raises well-known issues related to the big interface problem [7].

Filtering to the Different Platforms was an approach that uses an intermediary microservice between other microservices and consumers, similar to a gateway microservice. This intermediary microservice has the sole purpose of filtering the outputs for specific platforms. The case described by the participant had three different platforms, namely mobile, web, and desktop, that need customized information from the same microservice. For example, when a consumer uses a mobile platform, the user interface returns more targeted and lean responses to these devices. That is, it was a filter of the original outputs from target microservice interface with its customization for the different platforms. This pattern seems to

be a form of resolution to mitigate problems from the previously presented approach.

In addition to the approaches to deal with the customization that emerged after the migration process to the microservice architecture, the interviewed participants were also inquired about integration between extracted microservices and the legacy system. We observed that, in five cases, where the migration process was completed by the interview date, four of them extracted the microservices in an incremental process. That is, some microservices were extracted and integrated with the legacy system and the resulting behavior was observed. In this integration, it was common to use feature toggles or similar mechanisms to switch between the legacy system and the integration between extracted microservices. A feature toggle basically is a variable used in a conditional statement to guard code blocks, with the aim of either enabling or disabling the feature code in those blocks for testing or release [4, 25]. This was used in a production environment for the problems reported or observed by the team monitoring this transition. This approach allowed a fast return to a stable version (legacy system). In general, some microservices with database access (usually a key-value database) was used as a filter to choose the switch between using only the legacy system and the extracted microservices integrated with the legacy system. One of the interviews reported the use of this switching strategy between both versions for a small portion of their users. 43% of the survey respondents reported the employment of specific steps related to the integration between the legacy system and the extracted microservices. Besides that, the effort required in most cases was medium or high.

> ***RQ2.2:*** *Three approaches employed to implement the customization required after the microservices extraction are (i) Copy and Paste, (ii) Big Interface and (iii) Filtering to the Different Platforms. Microservices are usually extracted in an incremental process. Moreover, feature toggles (or similar mechanisms) are used to switch between the legacy system and the extracted microservices integrated with the legacy system.*

5 THREATS TO VALIDITY

The first threat is related to the number of questions in the survey, which might discourage the subjects' participation. To alleviate this threat we conducted an expert review with two specialists [19]. After considering their feedback, we conducted a pilot study with four subjects. In this pilot, we observed an acceptable participation rate.

Another threat concerns the number of valid respondents in the survey. This sampling process resulted in the recruitment of 90 individuals. A participation rate of 29% is considered good for online surveys of this kind, which usually ranges from 3% to 10%. In addition, most of the survey participants declared to have significant experience in migrating systems to microservice architecture (see Section 4.1). The quality of subjects was reached due to the fact we followed a formal recruitment strategy (see Section 3.2).

As far as the interview is concerned, a threat affects the process of collecting data during the interviews. To deal with this threat we asked the participants permission to record the interviews for future analysis and transcriptions. In addition, all interviews were performed by an interviewer, which presented the questions; and a scribe responsible for taking notes, analyzing the respondent behavior, and asking additional questions.

6 RELATED WORK

Francesco *et al.* [15] interviewed and applied a questionnaire to developers. Their goal was to understand the performed activities, and the challenges faced during the migration. They reported what are the existing system artifacts (e.g., source code and documents) the respondents used to support the migration. The main reported challenges were: (i) the high level of coupling, (ii) the difficulty of identifying the service boundaries, and (iii) the microservices decomposition. However, they did not specifically analyze the usefulness of variability criterion addressed in our survey.

Taibi *et al.* [27] also conducted a survey with the objective of elucidating motivations that led to the microservices migration process and what were the expected returns. The main motivations were the improvement of maintainability, scalability, and delegation of team responsibilities. In addition, difficulties were cited in this process, such as decoupling from the monolithic system, followed by migration, and splitting of data in legacy databases.

7 CONCLUSIONS

This paper presented an exploratory study composed of two phases. In the first phase, a survey was applied to specialists experienced in the migration process to microservice architectures. In this survey, we also inquired about the usefulness of variability and implementation mechanisms. In the second phase, we asked an interview with survey participants. We asked about customization requests after the microservices extraction, and the mechanisms used by the participants to deal with post-migration customization.

The results revealed that half of the participants dealt with systems with variability during the migration. Among these practitioners, two-third considered variability as useful or very useful to extract of microservices. We also observed initial evidence that microservices extraction can increase software customization, mainly because some users requested for customization after the microservices extraction. The most common approaches to implement customization in extracted microservices are copy and paste, big interface, and filtering to different platforms. Moreover, feature toggles (or similar mechanisms) are commonly used to switch/integrate the legacy system and the extracted microservices.

Our study is still ongoing. So, for future work we expect to have more responses to our survey and more interviewees. In this way, we will be able to draw additional analyses on other useful criteria for extracting configurable microservices from monolithic legacy systems.

ACKNOWLEDGMENTS

This research was funded by CNPq grants 434969/2018-4, 312149/2016-6, 408356/2018-9 and 428994/2018-0, CAPES grant 175956, FAPERJ grant 22520-7/2016.

REFERENCES

[1] N. Alshuqayran, N. Ali, and R. Evans. 2016. A Systematic Mapping Study in Microservice Architecture. In *International Conference on Service-Oriented Computing and Applications (SOCA)*. 44–51.

[2] Sven Apel, Don Batory, Christian Kstner, and Gunter Saake. 2013. *Feature-Oriented Software Product Lines: Concepts and Implementation*. Springer Publishing Company, Incorporated.

[3] Felix Bachmann and Paul C Clements. 2005. *Variability in software product lines*. Technical Report CMU/SEI-2005-TR-012. Carnegie Mellon University - Software Engineering Institute.

[4] Len Bass, Ingo Weber, and Liming Zhu. 2015. *DevOps: A software architect's perspective*. Addison-Wesley Professional.

[5] Jesús Bisbal, Deirdre Lawless, Bing Wu, and Jane Grimson. 1999. Legacy Information Systems: Issues and Directions. *IEEE Software* 16, 5 (Sept. 1999), 103–111.

[6] A. Bucchiarone, N. Dragoni, S. Dustdar, S. T. Larsen, and M. Mazzara. 2018. From Monolithic to Microservices: An Experience Report from the Banking Domain. *IEEE Software* 35, 3 (2018), 50–55.

[7] Robert C. Martin. 2002. *Agile Software Development, Principles, Patterns, and Practices* (1st ed.). Pearson.

[8] Rafael Capilla, Jan Bosch, and Kyo-Chul Kang. 2013. *Systems and Software Variability Management: Concepts, Tools and Experiences*. Springer Publishing Company, Incorporated.

[9] Luiz Carvalho, Alessandro Garcia, Wesley K. G. Assunção, Rafael de Mello, and Maria Julia de Lima. 2019. Analysis of the Criteria Adopted in Industry to Extract Microservices. In *International Workshop on Conducting Empirical Studies in Industry and International Workshop on Software Engineering Research and Industrial Practice (CESSER-IP)*. IEEE Press, Piscataway, NJ, USA, 22–29.

[10] Frederick Chong and Gianpaolo Carraro. 2006. Architecture strategies for catching the long tail. *MSDN Library, Microsoft Corporation* (2006), 9–10.

[11] Nicola Dragoni, Saverio Giallorenzo, Alberto Lluch Lafuente, Manuel Mazzara, Fabrizio Montesi, Ruslan Mustafin, and Larisa Safina. 2017. *Microservices: Yesterday, Today, and Tomorrow*. Springer International Publishing, Cham, 195–216.

[12] Yael Dubinsky, Julia Rubin, Thorsten Berger, Slawomir Duszynski, Martin Becker, and Krzysztof Czarnecki. 2013. An exploratory study of cloning in industrial software product lines. In *European Conference on Software Maintenance and Reengineering (CSMR)*. IEEE, 25–34.

[13] D. Escobar, D. Cárdenas, R. Amarillo, E. Castro, K. Garcés, C. Parra, and R. Casallas. 2016. Towards the understanding and evolution of monolithic applications as microservices. In *XLII Latin American Computing Conference (CLEI)*. 1–11.

[14] Susan Fowler. 2016. *Production-Ready Microservices* (1st ed.). O'Reilly Media.

[15] P. Di Francesco, P. Lago, and I. Malavolta. 2018. Migrating Towards Microservice Architectures: An Industrial Survey. In *International Conference on Software Architecture (ICSA)*. 29–2909.

[16] Paolo Di Francesco, Patricia Lago, and Ivano Malavolta. 2019. Architecting with microservices: A systematic mapping study. *Journal of Systems and Software* 150 (2019), 77 – 97.

[17] J. Gouigoux and D. Tamzalit. 2017. From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture. In *International Conference on Software Architecture Workshops (ICSAW)*. 62–65.

[18] W. Jin, T. Liu, Q. Zheng, D. Cui, and Y. Cai. 2018. Functionality-Oriented Microservice Extraction Based on Execution Trace Clustering. In *International Conference on Web Services (ICWS)*. 211–218.

[19] Johan Linaker, Sardar Muhammad Sulaman, Rafael Maiani de Mello, Martin Höst, and Per Runeson. 2015. Guidelines for Conducting Surveys in Software Engineering. (2015).

[20] Welder Luz, Everton Agilar, Marcos César de Oliveira, Carlos Eduardo R. de Melo, Gustavo Pinto, and Rodrigo Bonifácio. 2018. An Experience Report on the Adoption of Microservices in Three Brazilian Government Institutions. In *XXXII Brazilian Symposium on Software Engineering (SBES)*. ACM, New York, NY, USA, 32–41.

[21] G. Mazlami, J. Cito, and P. Leitner. 2017. Extraction of Microservices from Monolithic Software Architectures. In *International Conference on Web Services (ICWS)*. 524–531.

[22] Ralph Mietzner, Andreas Metzger, Frank Leymann, and Klaus Pohl. 2009. Variability Modeling to Support Customization and Deployment of Multi-tenant-aware Software As a Service Applications. In *Workshop on Principles of Engineering Service Oriented Systems (PESOS)*. IEEE Computer Society, Washington, DC, USA, 18–25.

[23] Sam Newman. 2015. *Building Microservices* (1st ed.). O'Reilly Media.

[24] Claus Pahl and Pooyan Jamshidi. 2016. Microservices: A Systematic Mapping Study. In *International Conference on Cloud Computing and Services Science (CLOSER)*. 137–146.

[25] Md Tajmilur Rahman, Louis-Philippe Querel, Peter C. Rigby, and Bram Adams. 2016. Feature Toggles: Practitioner Practices and a Case Study. In *13th International Conference on Mining Software Repositories (MSR)*. ACM, New York, NY, USA, 201–211.

[26] C. B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering* 25, 4 (July 1999), 557–572.

[27] D. Taibi, V. Lenarduzzi, and C. Pahl. 2017. Processes, Motivations, and Issues for Migrating to Microservices Architectures: An Empirical Investigation. *IEEE Cloud Computing* 4, 5 (2017), 22–32.

[28] Leonardo P. Tizzei, Marcelo Nery, Vinícius C. V. B. Segura, and Renato F. G. Cerqueira. 2017. Using Microservices and Software Product Line Engineering to Support Reuse of Evolving Multi-tenant SaaS. In *21st International Systems and Software Product Line Conference (SPLC)*. ACM, New York, NY, USA, 205–214.

[29] Alexander von Rhein, Alexander Grebhahn, Sven Apel, Norbert Siegmund, Dirk Beyer, and Thorsten Berger. 2015. Presence-condition Simplification in Highly Configurable Systems. In *37th International Conference on Software Engineering (ICSE)*. IEEE Press, Piscataway, NJ, USA, 178–188.

[30] Coburn Watson, Scott Emmons, and Brendan Gregg. 2015. *A Microscope on Microservices*. http://techblog.netflix.com/2015/02/a-microscope-on-microservices.html

[31] Claes Wohlin. 2014. Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering. In *International Conference on Evaluation and Assessment in Software Engineering (EASE)*. ACM, New York, NY, USA, Article 38, 10 pages.