Institute of Software Technology

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Exploring and Categorizing Maintainability Assurance Research for Service- and Microservice-Based Systems

Adrian Weller

Course of Study:	B.Sc. Softwaretechnik
Examiner:	Prof. Dr. Stefan Wagner
Supervisor:	Justus Bogner, M.Sc.
Commenced:	December 5, 2018
Completed:	June 5, 2019

Kurzfassung

Im Laufe des Softwarelebenszyklus eines Programms innerhalb einer sich ständig wechselnden Softwareumgebung ist es wahrscheinlich, dass dieses Programm regelmäßig gewartet werden muss. Wartungen kosten Geld und somit ist es wichtig, dass ebensolche Wartungen effizient und effektiv durchgeführt werden können. Im Laufe der Geschichte der Softwareentwicklung traten unter anderem zwei Architekturmuster hervor: Serviceorientierte Architektur und Microservices. Da diese Architekturmuster ein hohes Maß an Wartbarkeit versprechen, wurden viele Altsysteme hin zu diesen modernen Architekturen migriert. Es kann fatale Folgen für Unternehmen haben, wenn Änderungen an einem System nicht schnell, risikofrei und fehlerfrei umgesetzt werden können. Es wurden bereits viele Forschungsarbeiten bezogen auf die Wartbarkeit von serviceorientierter Architektur publiziert. Systeme basierend auf Microservices fanden jedoch, bezogen auf Wartbarkeitssicherung, nicht viel Beachtung. Sämtliche Forschungsarbeiten befinden sich verteilt auf viele Literaturdatenbanken, wodurch ein umfassender Überblick erschwert wird. Um einen solchen Überblick bereitzustellen, führten wir im Rahmen dieser Bachelorarbeit eine systematische Literaturstudie durch, die sich mit der Wartbarkeitssicherung von serviceorienter Architektur und Systemen basierend auf Microservices beschäftigt.

Zur Durchführung dieser systematischen Literaturstudie entwickelten wir eine Reihe von relevanten Forschungsfragen sowie ein striktes Forschungsprotokoll. Aufbauend auf diesem Protokoll sammelten wir insgesamt 223 Forschungsarbeiten von verschiedenen Herausgebern. Diese Arbeiten wurden bezüglich ihres Inhalts zuerst in drei Gruppen von Kategorien unterteilt (architektonisch, thematisch und methodisch). Danach wurden die jeweils relevantesten Forschungsrichtungen aus jeder thematischen Kategorie herausgearbeitet und vorgestellt. Zum Abschluss wurden deutliche Unterschiede der in den Forschungsarbeiten präsentierten Inhalte in Bezug auf serviceorientierte Architektur und Microservice-basierte Systeme herausgearbeitet und dargestellt.

Unsere Ergebnisse zeigten eine deutliche Unterrepräsentation von Forschungsarbeiten zur Wartbarkeitssicherung für Microservice-basierte Systeme. Während der Untersuchung der Kategorien konnten wir diverse Forschungsrichtungen innerhalb dieser feststellen. Ein Beispiel hierfür ist die Forschungsrichtung “change impact in business processes” in der Kategorie “Change Impact and Scenarios”. Abschließend konnten wir einige Unterschiede bezogen auf die gesammelten Forschungsarbeiten zwischen Systemen basierend auf einer serviceorientierten Architektur und Systemen basierend auf Microservices feststellen. Ein solcher Unterschied kann zum Beispiel in der Kategorie “Antipatterns and Bad Smells” gefunden werden. Im Vergleich zu Forschungsarbeiten, welche sich auf serviceorientierte Architektur beziehen, beinhalten Forschungsarbeiten im Zusammenhang mit Systemen auf Basis von Microservices nur grundlegende Informationen zu Antipatterns, jedoch keine Herangehensweisen, um diese zu erkennen.

Aufgrund unserer Ergebnisse schlagen wir einen stärkeren Fokus auf Forschung zur Wartbarkeitssicherung in Microservice-basierten Systemen vor. Mögliche zukünftige Forschungsarbeiten könnten überprüfen, ob Herangehensweisen zur Wartbarkeitssicherung von serviceorientierter Architektur auch bei Microservices anwendbar sind. Darüber hinaus schlagen wir die Durchführung von systematischen Literaturstudien vor, welche Themen wie “runtime adaptation”, “testing” und “legacy migration” untersuchen, da diese Themen in unserer Literaturstudie ausgeschlossen wurden.

Abstract

It is very likely that software running in an everchanging environment needs to evolve at multiple points during its lifecycle. Because maintenance costs money, it is important for such tasks to be as effective and efficient as possible. During the history of software development service- and microservice-based architectures have emerged among other architectures. Since these architectures promise to provide a high maintainability, many legacy systems are or were migrated towards a service- or microservice-based architecture. In order to keep such systems running, maintenance is inevitable. While a lot of research has been published regarding maintainability assurance for service-based systems, microservice-based systems have not gotten a lot of attention. All published research is spread across several scientific databases which makes it difficult to get an extensive overview of existing work. In order to provide such overview of maintainability assurance regarding service- and microservice-based systems, we conducted a systematic literature review.

To support our literature review, we developed a set of meaningful research questions and a rigid research protocol. Based on our protocol we collected a set of 223 different papers. These papers were first categorized into a threefold set of categories (architectural, thematical and methodical). After that, the most relevant research directions from each thematical category were extracted and presented. Lastly, we extracted and presented notable differences between approaches relating to service-oriented architecture or microservice-based systems.

Our findings show a clear underrepresentation of maintainability assurance approaches suitable for microservice-based systems. We further discovered that regarding our formed categories, we could find several research directions such as change impact in business processes in “Change Impact and Scenarios”. In the end, we could identify some differences between service- and microservice-based systems concerning approaches we retrieved in this thesis. A difference, for example was that in comparison with papers related to service-oriented architecture in “Antipatterns and Bad Smells”, microservices related papers only contained basic information on antipatterns, but no approaches to detect them.

Due to our findings we suggest a higher participation in research regarding maintainability assurance for microservice-based systems. Possible future work in this area could include further research on the applicability of service-oriented maintainability assurance approaches or techniques in microservice-based systems. Furthermore, future researchers could conduct follow-up literature reviews and investigate topics such as runtime adaptation, testing and legacy migration, since we excluded such topics from this thesis.

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interface. 47

BPEL Business Process Execution Language. 45

BPMN Business Process Model and Notation. 45

ESB Enterprise Service Bus. 17

HTTP HyperText Transfer Protocol. 18

MDSD Model Driven Software Development. 37

OOP Object Oriented Programming. 47

QMOOD Quality Model of Object Oriented Design. 50

REST Representational State Transfer. 18

SBE Scenario Based Evaluation. 38

SBS Service-Based Systems. 25

SD Service Decomposition. 37

SI Service Identification. 37

SIM Service Identification Method. 26

SLR Systematic Literature Review. 15

SOA Service-Oriented Architecture. 15

SQA Software Quality Assurance. 22

WADL Web Application Description Language. 51

WSDL Web Services Description Language. 17

XML eXtensible Markup Language. 17

1 Introduction

In this chapter, we will take a short look at what motivates this thesis, at the objectives we intend to solve and the remainder of this paper.

1.1 Motivation

With fast changing markets, it is likely that software running in this environment needs to evolve in order to keep it working. Since maintenance is basically development [Wag13, p.11] and therefore costs money, it is highly desirable to make this process as cheap and effective as possible. Throughout the history of software development, several architectural styles have emerged and were incorporated by developers in future software systems. One of these architectural styles is the well established Service-Oriented Architecture (SOA). Another, rather new architectural style is the microservices architecture. Both approaches promise to provide good maintainability and due to this, many systems are or were migrated towards one of the mentioned architectures. In order to keep software based on SOA or microservices running for a long period of time, it is inevitable that a potential software system gets maintained (often several times) through its lifespan. Since the emerging of SOA and microservices, a lot of research has been published related to maintainability assurance. The problem with most research is that it is spread across several databases and books or journals. Therefore, it is difficult to quickly gain an overview of existing scientific approaches and techniques. To solve this problem, the approach of conducting a Systematic Literature Review (SLR) exists. When done rightly, a researcher conducting an SLR is capable of gathering knowledge through multiple sources, providing an overview of the work which has been done and by this also guiding future work through revealing existing research gaps. However, to the best of our knowledge, no one has conducted an SLR to provide a summary of existing scientific research related to maintainability assurance in SOA and microservices until today. It is the goal of this thesis to close this gap and provide an extensive overview of maintainability assurance approaches and techniques for service- and microservice-based systems. Providing this overview benefits academia through revealing possible areas of future work as well as industry, with providing maintenance approaches that can help to maintain a system more efficiently.

1.2 Objective

As already mentioned, it is the intention of this thesis to provide an extensive overview on maintainability assurance for SOA and microservice-based systems. In order to achieve this goal, we provide an SLR which answers the following research questions:

1. **RQ1**: How can maintainability assurance approaches and techniques for service- and microservice-based systems that have been proposed in scientific literature be categorized?

2. **RQ2**: How are the identified publications distributed among the formed categories?
3. **RQ3**: What are the most relevant research directions, challenges or gaps per identified category?
4. **RQ4**: Are there notable differences between the approaches and techniques proposed for service-based systems and those for microservices?

1.3 Remainder

The remainder of this paper is organized as follows: **chapter 2** provides technical background information for important concepts, used in this thesis. **Chapter 3** gives a brief review and summary on related work in the area of this thesis. **Chapter 4's** content is twofold. Firstly, an overview of the general approach of an SLR is provided, then the research protocol specifically tailored to this study is shown. **Chapter 5** presents the results of the study. Firstly, more general results will be presented and after this, the research questions will be answered and discussed. **Chapter 6** provides limitations of this study and possible threats to validity. **Chapter 7** concludes the study with a summary, our implications and an outlook on possible future work.

2 Technical Background

This chapter provides an overview of important terms and concepts which are needed as a basis to fully understand this study.

2.1 Service-Oriented Architecture

SOA is an architectural style that can be seen as an approach of combining different software services, which communicate to other services through predefined protocols, to form an (in theory) easily scalable and maintainable software system.

2.1.1 What is a Service?

A service is a software program, which provides a specific functionality and is stateless, not context specific and self-contained. However, this is a very broad definition and applies to a lot of programs existing. There is more to a service which we will elaborate now.

2.1.2 Service Characteristics

Services usually consist of three main components [Mul]: 1) an interface, 2) a contract and 3) the specific implementation of the service. The interface is the component available to a service requester, it defines how the requests are performed. The service contract defines how interaction between consumer and requester must happen. Lastly, the implementation is the actual source code of a service, which defines the functionality of a service. Services in SOA have three characteristics [Pap03]:

1. Technology neutral: services should be invocable through standardized technology which is available to most IT environments.
2. Loosely coupled: a service must not require knowledge, internal structures or conventions of other services.
3. Support of location transparency: definition and location of a service should be stored in a repository.

To describe a web service, Web Services Description Language (WSDL) is used. WSDL is a language which is written in eXtensible Markup Language (XML) and able to specify the location and the methods of the service [w3sd]. In SOA, there are several types of services which can be divided in two categories: business services and infrastructure services [Mul]. Business services are responsible for performing business processes like sending an invoice to a customer or providing

a billing service. Infrastructure services, however, exist to provide technical functionality which is needed to execute business processes. Often these infrastructure services are managed centrally through an Enterprise Service Bus (ESB).

2.1.3 Service Communication

In order to enable services to communicate with each other, a way of communication needs to be provided.

Enterprise Service Bus

As promoted by Kress et al. in [KMN+13], the ESB can be perceived as a middleware solution which supports the interoperability of services in an heterogeneous environment.

Communication Protocols

Whether using an ESB or not, services need to follow communication protocols in order to exchange data between each other.

XML XML is a language which was designed to be read both by humans and machines and is used to transport and store data [w3sc].

SOAP The term SOAP stands for Simple Object Access Protocol, and is an application communication protocol based on XML for sending and receiving messages. It is an envelope which identifies an XML document as a SOAP message [w3sb].

REST Representational State Transfer (REST) can be seen as a structural design approach using the HyperText Transfer Protocol (HTTP) requests GET, POST, DELETE and PUT to transfer data between services [w3sa].

2.1.4 SOA Relationship

At this point, it is valid to think that SOA is all about the communication between different, technology neutral and loosely coupled services.
However, it is important to say that SOA is not an architecture that is only about the services communicating (interacting) with each other. One can see SOA as an relationship [Pap03] between three types of participants, which are:

1. Service provider: handles definition of service description and publication to discovery agency.
2. Service discovery agency: publishes a service and makes it discoverable.

3. Service requestor (client): retrieves the service description by using the find operation from the service discovery agency and invokes the service (or interacts with the provided implementation.

Interactions between these participants incorporate the *publish*, *find* and *bind* operations as also can be seen in figure 2.1.

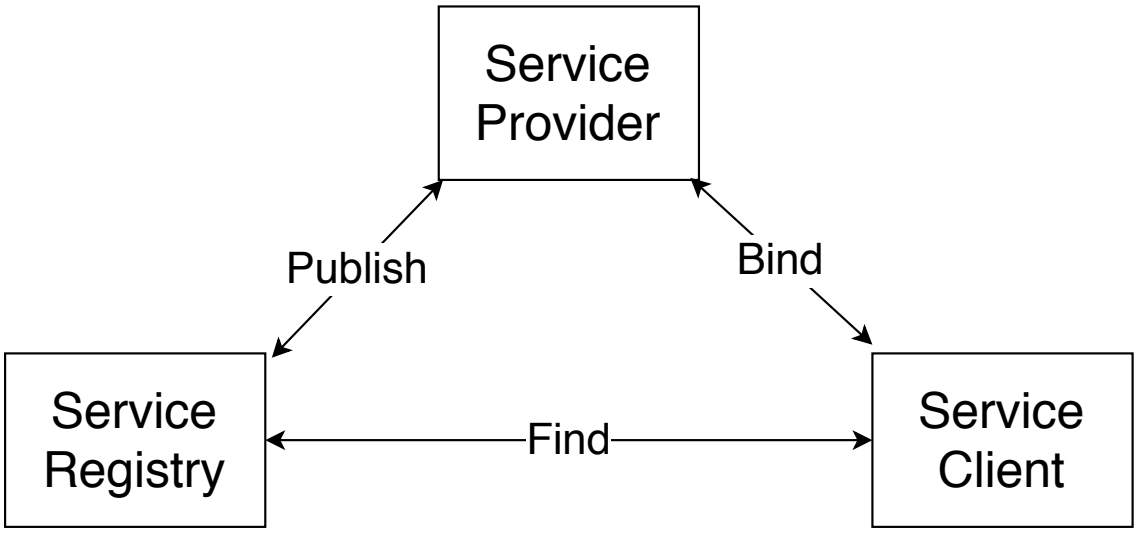

Figure 2.1: Service relationship

2.2 Microservices

The microservice-based architecture facilitates the usage of small service components (which are technology neutral, independently running programs).

2.2.1 Origin

The origin of the term microservices is not specified. However, it was first discussed at a workshop in Venice in 2011 [Fowa]. It can be seen as an architecture incorporating the characteristics mentioned in section 2.2.2.

2.2.2 Characteristics

Since there is no formal definition for the microservices approach, we will try to point out some common characteristics. Fowler states that not all microservice-based architectures share all characteristics but he expects them to exhibit most of the following characteristics [Fowa].

- Componentization via services: hereby a component is an independent unit of software.
- Organized around business capabilities: no siloed teams, preferable are cross-functional teams organized around capabilities.
- Products not Projects: this means a developer should accompany a system over the entire lifecycle (hereby the inspiration is taken from Amazon's "You build it, you run it"[1]).
- Smart endpoints and dumb pipes: in contradiction to the ESB, microservices favor the approach of not using a smart communication infrastructure between the services.
- Decentralized Governance: proposes the usage of the right tool (for example a specific programming language like Java or C++) for the right job.
- Decentralized Data Management: data should be stored in a decentralized way (so there is no central database), which means having different instances of the same database technology, or even using an entirely different database system for every service which is described as Polyglot Persistence [Fowb].
- Infrastructure Automation: Useful and applicable for Continuous Delivery or its precursor Continuous Integration. Another option is the usage for management in production.
- Design for failure: services may fail if a potential supplier is not available at the time of need and should therefore be designed to tolerate and handle failures appropriately.
- Evolutionary design: Microservices have great advantages in evolutionary design because only the redeployment of affected services is needed, which makes changes much easier and speeds up the process itself. However, this does not mean that a microservice-based architecture needs to undergo changes less frequently.

[1] Vogels, W. (2006) 'A Conversation with Werner Vogels', interviewed by Jim Gray for ACM QUEUE. Available at: https://queue.acm.org/detail.cfm?id=1142065 (Accessed: March 26, 2019)

2.2.3 Difference to SOA

SOA is perceived by Fowler ([Fowa]) to be many different things depending on the people incorporating this architectural style (e.g exposing software through web services or making a system communicate through some standart structure like XML) [Fow05]. This makes an actual comparison hard or even impossible. However, based on section 2.1, a few differences are perceivable at which we will have a closer look now. Firstly, services in SOA do not have specifications in terms of size while in a microservice-based architecture these services are kept as small as possible [Fowa]. Secondly, many SOA applications make usage of an ESB which is used as a mediator between the services (this could be seen as a smart pipe). In a microservices environment, however, only the endpoints (the services) should be smart in order to get services as decoupled as possible. Thirdly, while one of the microservices characteristics is decentralization [Fowa], for SOA, centralization is very important, as can be seen in [Erl09]. Fourthly, while SOA relies on service orchestration and service choreography, microservice-based architectures however, prefer service choreography over service orchestration. Furthermore, for a microservice-based architecture, service choreography should be minimized since this can lead to high efferent coupling [Ric16]. For this thesis we hold the assumption that even if microservices can be seen as a form of SOA, both of them are different architectural styles with shared commonalities.

2.3 Maintainability

In this section, we will define and explain basic terms relating to software maintenance and maintainability.

2.3.1 Software Maintenance

Wagner defines maintenance as the modification of software after its release. Maintenance can be carried out to correct faults, improve performance and other attributes or to adapt to a changing environment [Wag13, p.23].

Types of Maintenance

To fully understand how a system can be maintained, one must also look at the different types of maintenance, which we will clarify now in this section. There are four types of maintenance [Que]:

1. Perfective maintenance, which is about introducing new features to the software or adapting to new requirements of the system.
2. Adaptive maintenance is a type of maintenance, triggered by surrounding factors like a changing environment (e.g. new operating system or even new business rules).
3. Corrective maintenance addresses the repair of errors or faults in a software system.

4. Preventive maintenance, are mostly tasks corresponding to the optimization of a software product in the long run (e.g. increasing further maintainability or scalability), which can be, for example, restructuring code.

Software Quality Assurance

The goal of Software Quality Assurance (SQA) is to: 1) assess the adequacy of a process which is used to develop or modify software and 2) assess the adherence to this process [17, p.425].

2.3.2 Maintainability

Definition In direct relation to the definition of software maintenance in section 2.3.1, maintainability defines the degree of how easily and quickly a system can undergo these changes [17].

Maintainability Assurance In this thesis we define maintainability assurance as a set of activities that provide sufficient confidence that a software system can be sustainably maintained and evolved.

Maintainability Characteristics

Maintainability is represented by several characteristics at which we now take a brief look [ISO]:

- **Modularity** describes the degree of independence of the systems components. Preferred is a high modularity, so that a change in one component has an impact as minimal as possible on other components. In an SOA or microservices environment this means that the functionality of a system is provided through several distributed services, each one serving a specific purpose and only interacting with other services through interface calls. However, building software as a modular system does not directly guarantee high modularity since this can only be achieved through loose coupling, an important influencing factor we describe in section 2.3.3.
- **Reusability** is, as the term says, the degree to which a software component can be reused. This is not necessarily a key factor for SOA, since services can be reusable (agnostic) and not reusable (non-agnostic) [EW09]. However, striving for some level of reusability is essential for SOA [Sta18]. For microservices, it is preferable to reuse code by copy (rather than design reusable services), because this enables services to be further decoupled from each other [Sta18].
- **Analyzability** is about how easily a software engineer can understand a system and assess the change impact. For example, in case of a systems failure, analyzability is the degree of how easily a software engineer can assess which components need to be replaced. With regard to microservices and SOA we generally can say that analyzability depends on the size of a service and the overall amount of services. Smaller services make it easier to understand the functionality of a single service. The total amount of services influences, how easy a software engineer can understand the whole system since this makes it more complex.

- **Modifiability** declares how easily and quickly a system can be modified without getting into danger of introducing new defects. In an SOA or microservices environment, modifiability can be greatly affected by other services interacting with the service which is about to be changed (e.g. when service interfaces are modified). Microservices tend to have a higher modifiability through their evolutionary design characteristic briefly mentioned in section 2.2.2.
- **Testability** describes how easily and effectively a software product (or parts of it) can be tested, which includes establishment of test criteria and determining, which of those criteria have been met. For SOA this means that a single service is normally more easy to be tested than big monolithic systems. However, integration testing gets harder with the amount of overall services. In a microservices environment this is even more extreme since usually even more and even smaller services exist.

However, it is important to note that several retrieved papers in this SLR use other terms e.g. adaptability, which refers, in this case, more to functional adaptability than to the adaption to new hardware environments. We will elaborate this terms in section 2.3.2.

Further Terms

The following terms do not specifically refer to the ISO/IEC 25010 characteristics of maintainability, but are often used interchangeably when talking about maintenance or maintainability.

Evolvability Evolvability describes the degree of how easily a software system can accommodate future changes [BCL12b]. According to Bogner et al. in [BWZ17a], the extending and adaptive notion of maintainability is sometimes described with evolvability.

Adaptability Adaptability refers to the degree on how efficiently and effectively a system can be adapted to different environments (e.g. evolving hardware) [ISO]. However, when referring to adaptability or adaptation in this thesis, mostly functional adaptation is meant.

2.3.3 Influencing Factors

Many attributes can influence the maintainability of a system. In this section, we will list up several attributes and outline how they can influence maintainability.

Complexity Chowdhury et al. state that complexity is a design attribute often applied to the interaction between programmer and program and therefore depends on size, control structure and other factors [CZ11]. An increase in complexity therefore automatically decreases the maintainability through reducing modifiability and analyzability since high complexity nearly automatically hinders the comprehension of a program component.

Cohesion Cohesion is the degree of functional relationships between tasks in a software module [17]. In terms of maintainability a high degree of cohesion is wanted since it increases maintainability. High cohesion also often correlates to low coupling [Ing18, p.83].

Coupling As defined in [17], coupling describes (inter)dependency between software modules or routines. Low coupling hereby means a low dependency between modules or routines. This is desired since high coupling would mean that if a module is maintained it could have effects on other modules and therefore could reduce maintainability because in the worst case changes also need to happen in these modules.

3 Related Work

This chapter presents a brief overview of work related to this thesis.

As Soldani et al. found out when adopting microservices, one is also adopting the advantages (gains) and disadvantages (pains) of this architectural style [STH18]. In their paper, they conduct an SLR with grey literature on microservices. They identify, analyse and compare technical as well as operational pains and gains. They develop a taxonomy in which each identified pain and each identified gain is associated with a specific concern. Afterwards these concerns were assigned to common stages of the software lifecycle (design, development and operation). During their research, they also recognized that the microservices approach improves the maintainability due to the small size and self-containment of a service, which makes it more easy to understand for new personnel. However, the paper covers the entire pros and cons of microservices. It covers maintainability only as a positive result of the microservice-based architecture and not specifically how it can be achieved.

Breivold et al. conducted an SLR to get an overview of scientific papers, which thematize analysis and improvement of software evolvability on an architectural level [BCL12a]. They included 82 primary studies and summarized the studies into five main categories: quality considerations during design, architectural quality evaluation, economic evaluation, architectural knowledge management and modeling techniques. In their work they also focus on the aspect of maintainability since it is considered as a closely-related alternative term to evolvability by the authors. However, they do not focus on a specific set of architectures like SOA and microservices, which leads inevitably to the coverage of a very broad spectrum of architectural styles. Their paper differentiates itself greatly from this thesis, which proposes a more focused scope, covering only maintenance assurance approaches for SOA and microservices.

In their paper, Venters et al. provide an overview of software sustainability approaches but do not focus on Service-Based Systems (SBS) [VCB+18a]. Sustainability hereby is defined as a system's capacity to endure. The content of the paper is divided up into several chapters and they can be described as follows: sustainability from an architectural point of view, importance of sustainability for architectural decisions, software metrics for the estimation of sustainability with focus on a practical point of view, academic perspectives and issues and research challenges.

In their literature review, Bogner et al. present metrics from literature which can be used to automatically measure the maintainability of SOA software systems [BWZ17a]. Therefore they first provided a short overview of the history of maintainability metrics and later moved on, presenting a list of selected metrics specifically for SBS. At last, they investigated the applicability of these metrics to microservice-based systems. This paper is one of the few papers explicitly addressing microservices and how maintainability can be quantified with metrics within this architectural approach. However, it needs to be mentioned that this paper only covers a small area of maintainability assurance, i.e. it is not an alternative to this thesis.

[QLJL14] performed a systematic mapping study on regression testing techniques for web services. To achieve this, they conducted the mapping study on scientific work, which was released between 2000 and 2013 and studied 60 regression testing techniques in total from the perspective of different stakeholders (service developers, service providers, service registries, service integrators and service users). The paper focuses more on evaluating different regression testing techniques as on improving testability in the first place which also distinguishes it from the scope of our work.

Bani-Ismail et al. present a systematic literature review on service identification challenges in SOA [BB18a]. Challenges, identified by the authors were service quality attributes, business-IT alignment, systematic Service Identification Method (SIM), comprehensive SIM, tool support, validation, input artifact and configurability of SIM. An interesting finding was that service quality attributes (in particular service granularity) need more attention with future research as it is considered as a top challenge. The difference to our thesis is, that the paper uses an SLR to provide more information on service identification challenges, which is a much more detailed scope compared to our approach.

To provide more information on service-oriented system engineering challenges, Gu et al. conducted an SLR on scientific papers related to the topic, which were published between January 2000 and July 2008 [GL09]. An interesting fact which is provided by their tabular overview on challenges, is that maintenance is numerically not well represented. The paper uses an SLR to gain more knowledge on SOA but with notable different intentions compared to the work provided in this thesis.

In [VKG17], the authors conducted an SLR to analyze emerging standards, types of research conducted and the main practical motivations behind microservices related research. Deriving from the provided figures, one can say that nearly half of the research conducted, is related to the proposal of solutions. To get an answer to practical motivations behind microservices related research, the authors map the retrieved papers to different operational areas. An interesting, yet logical finding was that due to the relatively young architectural approach, most research papers focused on functionality and design questions. Furthermore, the authors analyzed emerging standards in microservice-based architectures and discovered that REST can be described as the standard for microservices. Looking at common tools, it is stated, that Docker can be seen as the most frequently used one. Compared to our work, the paper focuses more on giving a broad overview related to microservices research, not looking at specific aspects (e.g. maintainability) in detail.

In their paper, Alkharabsheh et al. provide a systematic mapping study concerning smell detection in software design, including research work from 2000 to 2017 [ACMT18]. They define bad smells as design defects, design flaws, anomalies, pitfalls, antipatterns and disharmonies. In contrast to this thesis, the paper focuses only on smell detection, but with a much broader scope concerning software architecture styles.

In conclusion, we can say that literature reviews with regard to maintainability assurance in SOA and microservice-based architectures are currently non-existent which proves the existence of a gap which we intend to close with this thesis.

4 Methodology

The following chapter will provide information about the chosen methodology and the developed research protocol.

4.1 Systematic Literature Review

As [KC07] stated in their guidelines, there are many reasons to conduct an SLR, which can be for example:

1. Summarization of existing work.
2. Identification of research gaps.
3. Based on found research gaps, suggesting research agendas to fill identified research gaps.

An SLR is a secondary study with the main goal of identifying and evaluating scientific work, related to an area of research. Therefore, to achieve this goal we proposed research questions we intended to answer. After completing this step we defined a research protocol capable of doing so. Firstly, we decided which data sources (electronic scientific databases) should be used. Secondly, we developed a reproducible search strategy, containing a consistent search term used through all data sources. After retrieving the first set of results we filtered all papers basing on our defined inclusion and exclusion criteria. With the now retrieved second result set, we conducted a forward citation search using google scholar and each new retrieved paper also was assessed using the before mentioned inclusion and exclusion criteria. After completing this step we categorized each paper (see also section 4.2.6). The full categorized result set can be inspected in appendix A and appendix B. The filtering steps performed on all results of the original search as well as on the result set of the forward citation search, the definition of categories and the assignment to those, were done independently by two researchers and differences of opinion were discussed. Lastly, we provided an answer to our proposed research questions.

4.2 Research Protocol

In this chapter we present how the SLR was carried out. We therefore firstly present our research questions and then our research strategy. A compact overview of our process can be seen in figure 4.1.

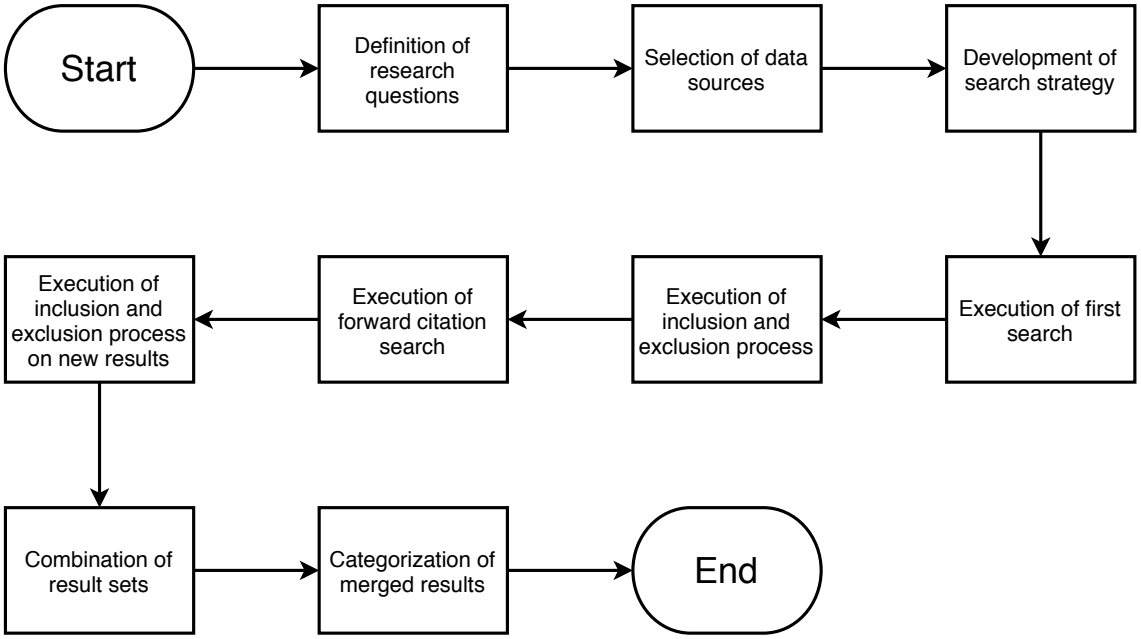

Figure 4.1: Research process

4.2.1 Research Questions

As we already mentioned in chapter 1.2, the following research questions will be answered:

1. **RQ1**: How can maintainability assurance approaches and techniques for service- and microservice-based systems that have been proposed in scientific literature be categorized?
2. **RQ2**: How are the identified publications distributed among the formed categories?
3. **RQ3**: What are the most relevant research directions, challenges or gaps per identified category?
4. **RQ4**: Are there notable differences between the approaches and techniques proposed for service-based systems and those for microservices?

4.2.2 Datasources

The following electronic databases were used during the basic research step:

1. IEEE Xplore
2. ACM Digital Library
3. ScienceDirect
4. SpringerLink

4.2.3 Search Strategy

To provide a reconstructable search, the searchterm was assembled as follows:

a) maintainability

b) evolution

c) modifiability

d) evolvability

1. service-oriented
2. service-based
3. SOA
4. microservice

(a OR b OR c OR d) AND (1 OR 2 OR 3 OR 4)

We didn't use proximity or wildcard search due to the lack of support from some of the search engines.
To narrow down and streamline results, there were several additions made depending on the search engine, since not every search engine is restricted to computer science or software engineering.

- ScienceDirect
  - Title, abstract or author specified keywords must contain: (software OR development) NOT (health OR psychology OR weather).
- Springer
  - Discipline: computer science
  - Language: English

4.2.4 Inclusion and Exclusion Process

To be included in the literature review, a paper needed to pass a three-step process which will be explained now in detail.

Basic Inclusion Criteria

In this first step, basic requirements were checked. To pass this step, all of the conditions below needed to be fulfilled.

1. Language was to be English.
2. Publishing date was to be between and including 2007 and 2018.

Because some engines provided an unmanageable amount of results, all results were filtered by relevance and enumerated from one to x. A paper was not considered if $x > 250$, to narrow down the result set to a reasonable size. There might be the possibility that some important papers were excluded this way but there also was a forward citation search. If a paper was relevant to a high degree, there was a good probability, that it was found while the citation search was conducted.

Content-Based Inclusion

The second step in itself was a two step process:

1. Title based inclusion

   To decide if a paper should be passed on to the next step, the following guidelines were developed:

   a) The title should have contained the search terms or modified versions/synonyms (e.g. a combination of "maintainability" and "service-oriented software" like in [PR11]).

   b) If this was not the case we investigated if hints existed that the topic was about microservices or service-based systems in combination with a maintainability related topic. Take for example: "Measuring the Quality of Service Oriented Design" ([SSP09]). It is clear that this scientific paper is about SOA. However, maintainability is not clearly mentioned in the title but quality attributes often include coupling and cohesion which can influence maintainability significantly. In such cases a paper was forwarded to step two (abstract-based inclusion). If a title could not be judged at all, the paper was also forwarded to step two. This is because sometimes titles were designed in a way we could not guess the content. Other researchers also have stumbled over this problem [DD08].

   c) If a paper could already be judged to be unsuitable by its title, it was excluded from the study right away. However, if it passed this step or the title could not be assessed, it (still) needed to pass the abstract-based inclusion.

2. Abstract-based inclusion

   If a paper was judged by its abstract, the following questions were asked:

   a) Are the authors specifically addressing microservices, SOA or related terms (e.g. SBS)? We excluded a paper if this was not the case like in [VCB+18b].

   b) Is a sufficient focus on maintainability (or related terms like evolvability or adaptability etc.) provided? A paper had to be excluded if nothing relating to maintainability was presented therein as in [BFH16].

⇒ If these questions could not be answered by only reading the abstract, other sections of the paper (e.g. conclusion) were skimmed.

While the following areas are in some way related to service-based maintainability assurance, we explicitly excluded such papers from the SLR. We wanted to narrow down the scope and provide a more detailed view of already incorporated topics since each of the following topics provide enough content to be dealt with in a separate SLR.

1. Runtime adaptation like in [FL13].
2. Testing like in [JGV+14].
3. Migration of legacy systems to microservices or SOA like in [LS07].

Quality-Based Exclusion

At last, a paper's quality was evaluated. If a paper had not met our expectations concerning quality it was excluded from this SLR. To be eligible for this thesis, the contribution of the paper or sub segment (this can be for example a single chapter) needed to be clearly described (this included also models, challenges, general conditions, limitations etc. if applicable like in [DMT13]).

4.2.5 Forward Citation Search

After conducting (and evaluating) the initial search, all included results were used for a forward citation search. To conduct this search step, we used Google Scholar, as this search engine is not restricted to a specific database and has a widespread range. For each accepted paper, a forward citation search was conducted and the first 15 results were retrieved for evaluation. The newly retrieved set of papers was now evaluated as described in chapter 4.2.4. However, if a paper was included via citation search, no additional citation search for the retrieved paper was conducted.

4.2.6 Categorization

In order to provide a sufficient categorization for each paper, all included papers were skimmed for their content and chosen approach (e.g. case study or SLR) and the first set of categories was proposed. Categorization is threefold: an architectural categorization, a thematical categorization and a methodical categorization. Each paper needed to be categorized into exactly one of the architectural categories and at least one of the thematical categories. A methodical categorization was optional. However, the first categorization was proven to be insufficient since too many papers needed to be listed as others. We redesigned the categories and came up with the current categorization which we present in section 5.2, where we also explain each set of categories briefly.

5 Results

In this chapter, we present the results from our SLR.

5.1 SLR Results

Before going into detail with answering the research questions, a short summary of general results is presented in this section.

5.1.1 Included Papers per stage

While conducting our research, we rigidly kept track on which paper was included or excluded in our process during a specific step. Figure 5.1 shows the exact amount of papers that were left after each step of our research protocol. We started with a total amount of 1000 papers distributed to four publishers. After the inclusion and exclusion process a total of 122 papers were left. We then conducted the forward citation search and the second inclusion and exclusion process. We merged the results and received a total of 223 papers.

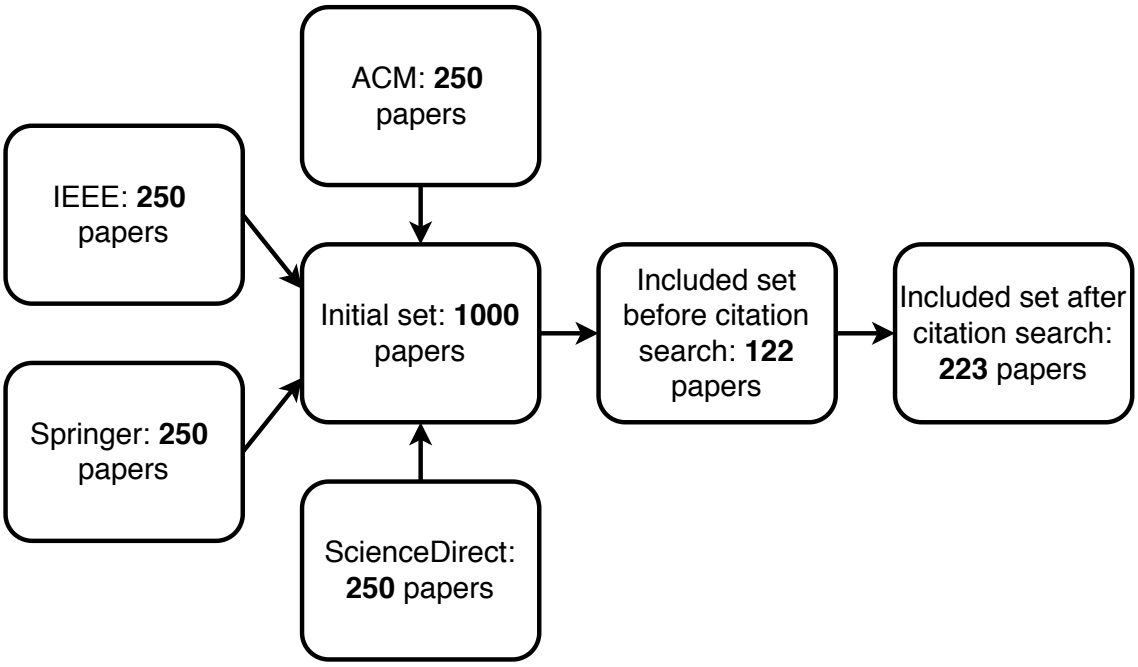

Figure 5.1: Included papers per stage

5.1.2 Distribution of Papers

In this section, we will take a closer look at how the papers were distributed for each step. Obviously the initial set was evenly distributed between ScienceDirect, Springer, ACM and IEEE. Figure 5.2 shows the distribution of papers before the citation search was conducted. An interesting finding here is that IEEE provided roughly half of all papers. The other half is represented by Springer, ACM and ScienceDirect, which provides the lowest amount of papers. Figure 5.3 displays the distribution of papers after the citation search was conducted. Just as before the citation search, IEEE provides roughly half of all papers included in this SLR. The other half is represented by ACM, Springer, ScienceDirect and various other electronic databases (e.g. publishers and universities), summarized as ”Other”. Figure 5.4 shows the distribution of all papers over the included years after the citation search. The diagram shows that there had been a varying release of papers over the years. The lowest point with only six releases can be found in 2007, which is not surprising because an SOA was pretty scarce at that time so there might have not been a lot of funding for researchers. It may also be due to the fact that SOA was relatively new and therefore nobody worried about maintenance or evolution so far. After 2007, there was a rise of released papers, peaking in 2011 with overall 35 papers released that year. The years after significant less papers were released with a local minimum in 2017 (15 papers).

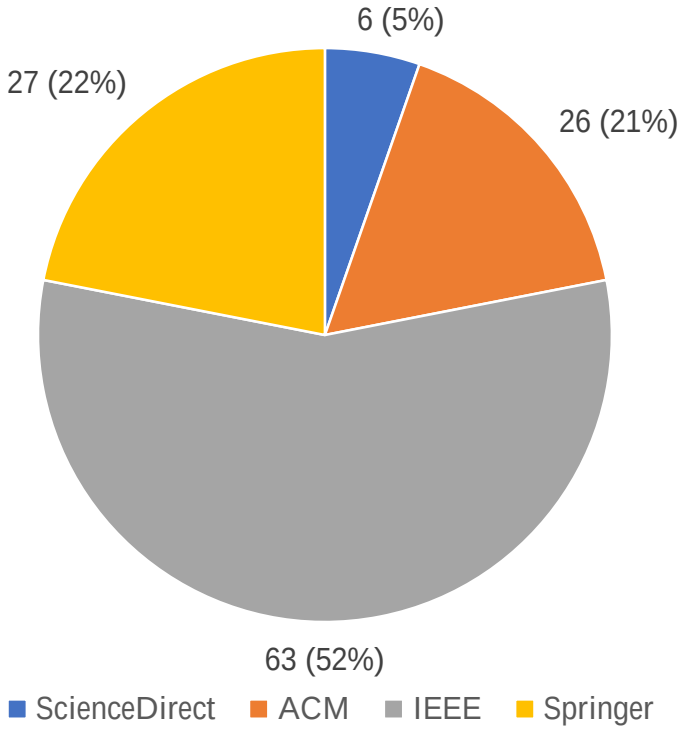

Figure 5.2: Distribution before citation search (per publisher)

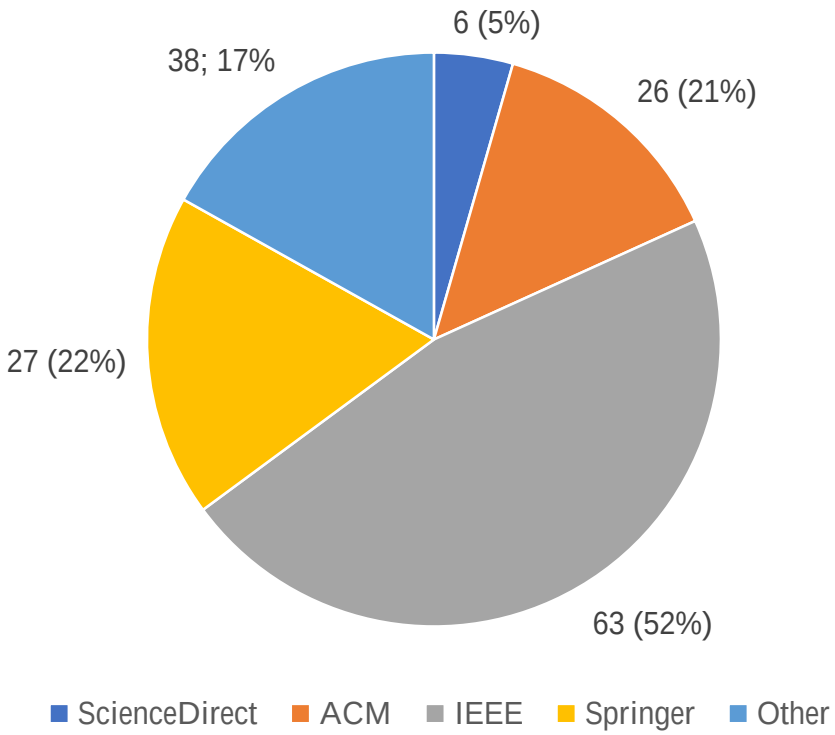

Figure 5.3: Distribution after citation search (per publisher)

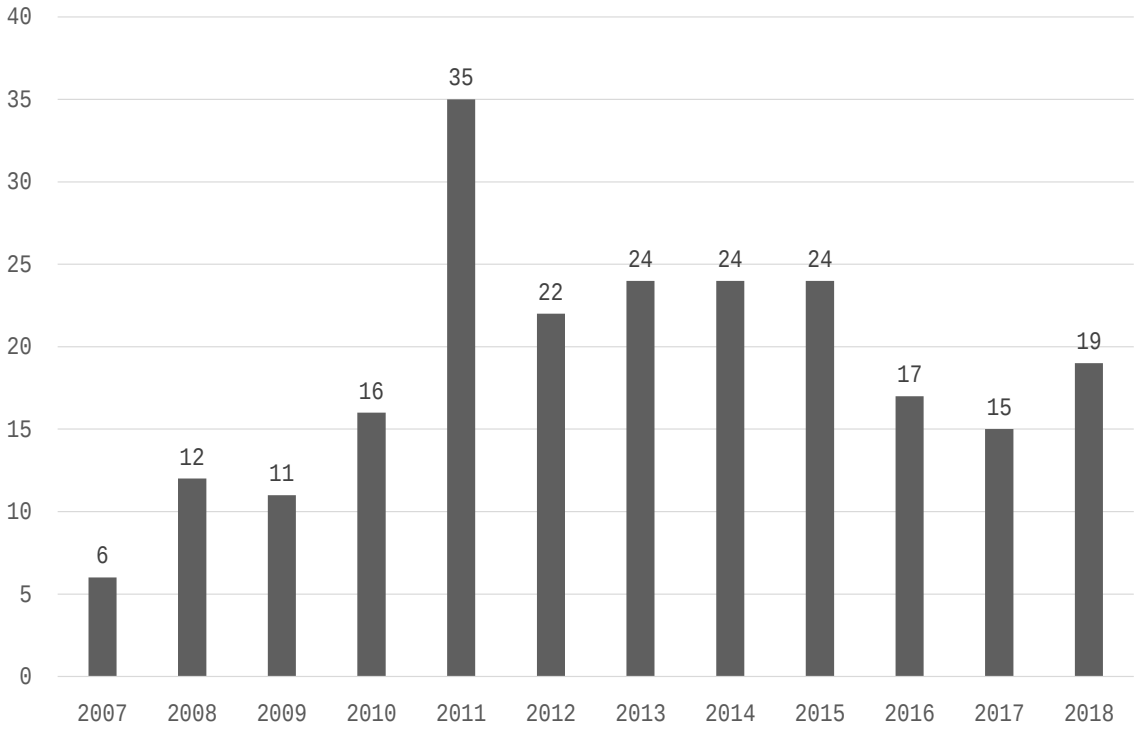

Figure 5.4: Overall distribution of publications (per year, after citation search)

5.2 Categories (RQ1)

In order to answer RQ1, we now present our suggested categorization approach. To get an overview of covered topics and methods used, we developed the following, threefold set of categories. Each set of categories is independent from the others. Every retrieved paper was assigned to exactly one architectural category and at least one thematical category. The third set of categories (methodical) however, was optional. In order to fully understand our categorization approach, we will first explain each set in general and then focus on the therein proposed categories. The full set of all categorized papers can be found in appendix A and in appendix B.

Table 5.1 gives a high-level overview on each set of categories.

Set	**Description**	**Multiple categories possible?**	**Mandatory**
Architectural	In this set papers were assigned to one of three categories based on their focus on either SOA, microservices or both.	No	Yes
Thematical	The thematical set incorporates nine categories which describe a paper's content on a topic-based level.	Yes	Yes
Methodical	This set contains the proposed method- and contribution-related categories. That means, a category can either be defined by therein presented approaches (e.g. literature study) or contributions (e.g. a reference architecture).	Yes	No

Table 5.1: Categorization

5.2.1 Architectural Categories

In this section, we present all categories which belong to the architectural categories set.

SOA

This category covers all papers whose main focus are SOA or relating topics (e.g. SBS) as in [Pal13].

Microservices

Analogous to the SOA category, this category covers all papers where the main focus is on microservices as in [ZNL17].

SOA and Microservices

If a paper's content was suitable for both SOA and microservices as in [BWZ17a], it was sorted into this architectural category.

5.2.2 Thematical Categories

In this section we present our nine thematical categories and explain them briefly.

Architecture Recovery and Documentation

In order to keep systems maintainable and evolvable, it is important to document systems well. However, mostly for older systems there is no documentation available which makes evolution hard or even impossible. This category contains papers which contribute to either providing a sufficient documentation of a system's architecture or providing information and approaches on how a system's architecture can be recovered (e.g. [KUSM18]).

Model-driven Approaches

The Model Driven Software Development (MDSD) approach refers to a style of software development which centers itself on building models of a system. These models are expressed as diagrams, for example with UML. The diagrams are used to specify a system for a modelling tool and then code is generated in a programming language [Fow08]. In terms of maintainability assurance, model-driven approaches can separate a system from its evolution and describe how this system can evolve [LCZ12]. An example for this can be seen in [LCZ12].

Patterns

This category includes papers related to patterns (such as [DMT13]). A pattern is a reusable solution (good practice) to a repeatedly presented issue in software design. However, a pattern is not a completed design, it is a description on how a component should be designed [MJ11]. Applying a pattern can improve a system's maintainability.

Antipatterns and Bad Smells

In contradiction to the patterns category, this category contains papers related to antipatterns and bad smells (e.g. [Pal13]). An antipattern (bad smell) can affect quality attributes such as maintainability in a negative way.

Service Identification and Decomposition

This category contains all papers related to Service Identification (SI) or Service Decomposition (SD) in service- and microservice-based architectures such as [Dam14]. SI and SD are closely related. SI is the process of deciding which functions and capabilities should be included into a service in service- and microservice-based systems. SD however, while it is a SI approach, focuses on splitting already existing services into smaller services with more distinct capabilities. The wrong execution of SI or SD can introduce antipatterns in a system's code. Possible antipatterns are

God Object Web Services (to much functionality) and Fine Grained Web Services (few low cohesive operations) [PMTG14]. Both antipatterns lower the overall cohesion of web services [PMTG14] and therefore also lower maintainability (see also section 2.3.3).

Change Impact and Scenarios

Scenario Based Evaluation (SBE) is the process of analyzing the impact of possible future events by also considering alternative outcomes, i.e., scenarios [Bal19]. SBE can be used to analyze software quality attributes (e.g. scalability, security, maintainability etc.). Since it is the goal this thesis to provide knowledge on maintainability assurance approaches and techniques, we only focus on SBE in combination with maintainability. A special case of SBE is change impact analysis. Change impact analysis focuses on the process of analyzing how changing one specific component can affect other components as well (e.g. through changing a service interface). If a paper related to either change impact analysis (e.g. [EMK12]) or SBE (as in [NGS15]), it was sorted into this category.

Maintainability Metrics and Prediction

Analogous to the more general term of software metrics, maintainability metrics are measures of quantifiable software characteristics relating to maintainability. To be in this category, a paper must present information about maintainability metrics or approaches on how these metrics can be used in order to predict maintainability related aspects in SOA or microservice-based systems such as in [DDD11].

Evolution Management

Evolution Management is the most general category beside “Other”. We sorted a paper into this category if therein presented approaches help to improve the overall evolution process (e.g. [BKC+10]). Possible improvements could be achieved through: 1) providing approaches to plan the evolution process, 2) speeding up the evolution process, 3) making the overall evolution process more resistant to faults or 4) eliminating potential negative consequences. However, when talking about eliminating negative consequences, it is important to differentiate it from change impact. We defined our understanding and scope of change impact in the corresponding section above. Every paper fitting our description above and relating to change impact but not focusing on change impact between different services or business processes, was sorted into this category.

Other

This category lists all papers which could not be sorted into one of the categories (e.g. [LWD08] or [DJC08]) above to avoid a large number of very small categories.

5.2.3 Methodical Categories

In this section, we present a brief overview of our methodical categorization. As already mentioned above, this categorization is optional and therefore not all papers are represented here.

Case Study, Field Study or Empirical Study

In this category, we included all papers containing a case study (e.g [KRN+11]), field study (e.g [KLS08]) or an empirical study (e.g [WKZ14]). We chose to include a paper having a case study, when therein presented approaches were demonstrated using a (mostly) exemplary system. If an approach was checked against a real world system (e.g. industry or Open Source) we included a paper for containing a field study. In order to be included for providing an empirical study, a paper must have presented results from a survey, interview or experiment.

Literature Study

All papers, in which the authors used a literature study (e.g. SLR, Systematic Mapping Study, etc.) as their mode of operation (as in [BB18a]) are mentioned here. We already explained the basic principles of an SLR in section 4.1.

Model or Taxonomy

This category presents all papers in which the authors present a model or taxonomy (as in [ZAB15]). In order to be identified as a taxonomy related paper, the authors must have presented a system which is able to organize their findings or research artifacts into groups with similar qualities (as in [PM15]) [Dicc]. To be identified as a model related paper, the authors needed to provide a representation of their idea or approach, which could be also in an abstracted way (i.e. as meta model) as in [IHL+13].

Processes and Methods

In processes and methods, all papers are contained which include specific methods (e.g [LMN10]) or processes (as in [WC15]) in order to improve and assure maintainability. A method is a used technique to achieve a goal [Dica], a process however, describes an entire sequence of activities to achieve a specific goal [Dicb].

Reference Architecture and Tools

In this category, we included all papers relating to the presentation or usage of reference architectures (as in [RGR11]) or tools (as in [WYZ11]). Reference architectures are reusable, generic templates for building system architectures [WB14a].

5.3 Distribution among Categories (RQ2)

In order to get a sufficient overview of the distribution among the formed categories, we first take a look at the distribution among the three architectural categories and then inspect the distribution of papers among the thematical as well as the methodical categories.

5.3.1 Distribution among Architectural Categories

Comparing all retrieved papers in figure 5.5 and adding up the “Microservices” and “Both” category, only 11% of all retrieved papers are related to microservices in our result set. The remaining 199 papers (89%) are related only to SOA. However, to compare the actual distribution on the topic of maintainability assurance, it would be more appropriate to consider only papers released in and after 2014. This is due to the fact that the first microservices related paper we retrieved, was published in 2014. Even if the naming of this architecture took place two years earlier [Fowa], microservices emerged from industry practice and therefore it took some time for microservices related topics to arrive in scientific publications. Looking at figure 5.6, we can see that roughly only a quarter of all released papers are related to microservices. In other words, only 2,5 out of 10 papers released, are related to maintainability assurance in microservice-based architectures. Recent research has shown that using a microservice-based architecture is perceived to have a positive impact on a software product’s maintainability [BFWZ19]. This might be a reason for the low amount of publications relating to maintainability assurance in microservice-based architectures.

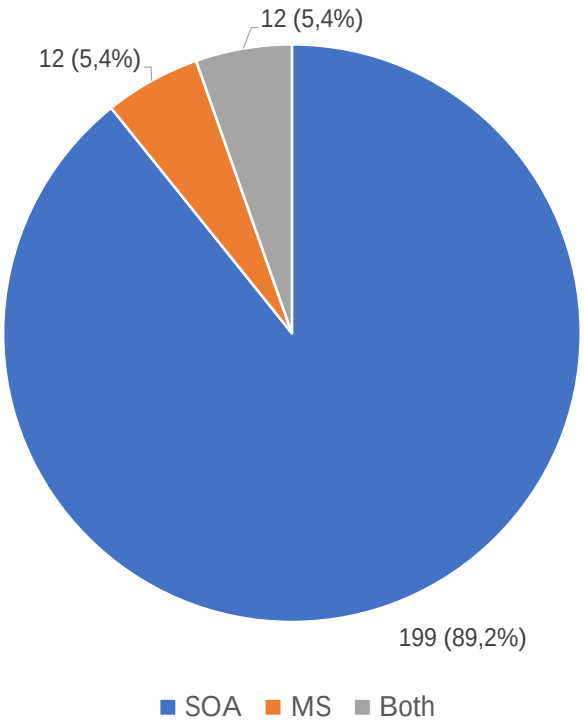

Figure 5.5: Papers related to SOA, microservices (MS) or both

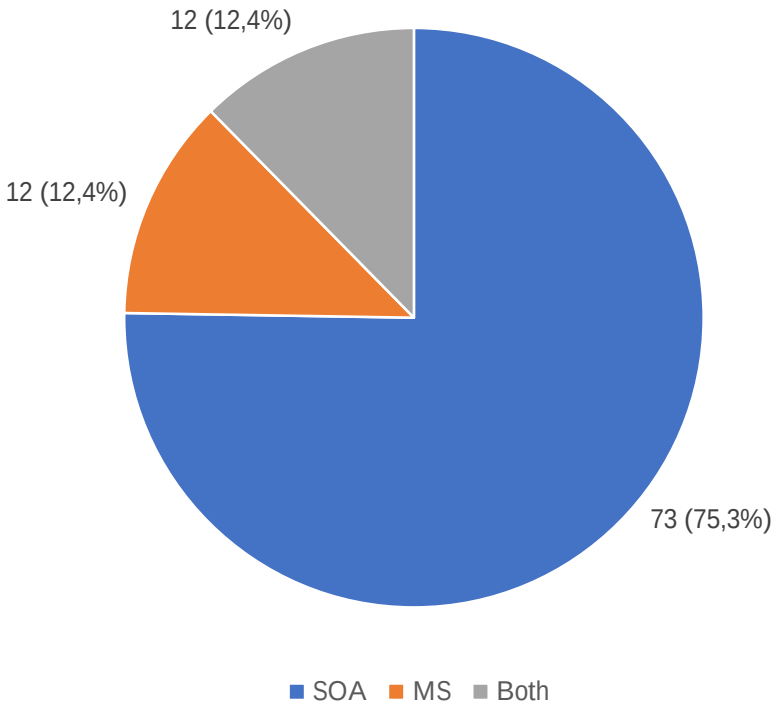

Figure 5.6: Papers related to SOA, microservices (MS) or both, 2014 until 2018

5.3.2 Distribution among Thematical Categories

Looking at the distribution among our formed thematical categories in figure 5.7, several things are ascertainable.

Firstly, 25,1% of all papers provide “Evolution Management” related content and therefore make it the best represented category shortly followed by “Maintainability Metrics and Prediction” (24,7%). Secondly, the three best represented categories “Evolution Management”, “Maintainability Metrics and Prediction” and “Change Impact and Scenarios”(17,5%), contain more than half of all collected approaches and techniques. Lastly, the other half consists of six smaller categories (all with less than 10%): “Service Identification and Decomposition” (9,4%), “Antipatterns and Bad Smells” (9,4%), “Other” (8,1%), “Patterns” (6,3%), “Model-driven Approaches” (5,4%) and “Architecture Recovery and Documentation” (4%).

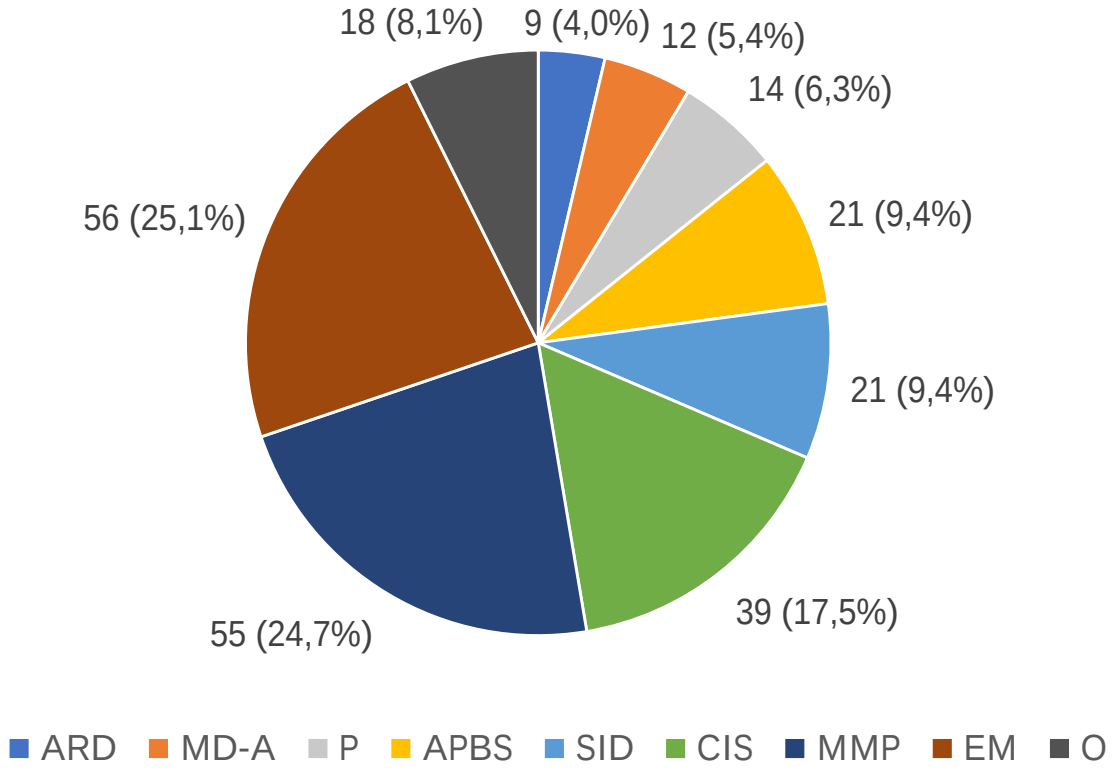

Figure 5.7: Distribution among thematical categories, all percentages relative to 223 papers

ARD = "Architecture Recovery and Documentation", **MD-A** = "Model-driven Approaches", **P** = "Patterns", **APBS** = "Antipatterns and Bad Smells", **SID** = "Service Identification and Decomposition", **CIS** = "Change Impact and Scenarios", **MMP** = "Maintainability Metrics and Prediction", **EM** = "Evolution Management", **O** = "Other"

5.3.3 Distribution among Methodical Categories

Noticeable in figure 5.8 is the fact that 141 papers (63,2%) included a case study, field study or empirical study. Containing 110 papers (49,3% of all papers) in total, "Processes and Methods" is the second largest methodical category. With 75 papers (33,6% of all papers) total, "Model or Taxonomy" is the third-largest category directly followed by "Reference Architecture and Tools" containing a total of 72 papers (32,3% of all papers). The least represented category is "Literature Study" with only 25 papers (11,2% of all papers).

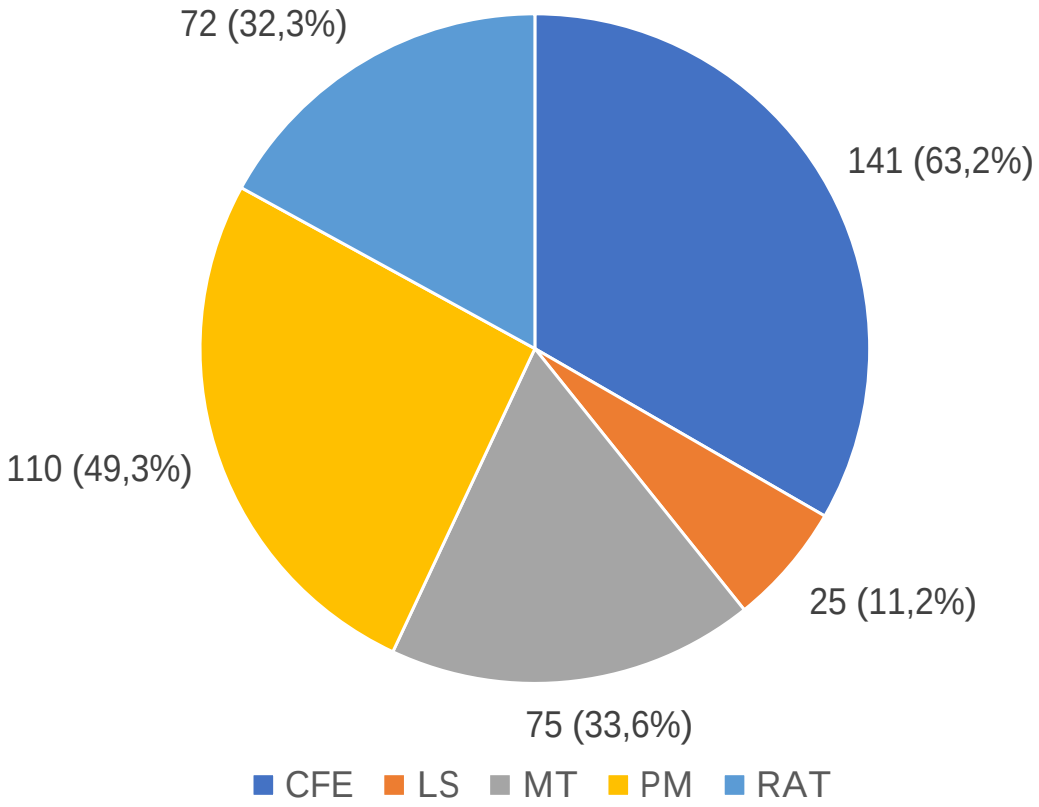

Figure 5.8: Distribution among methodical categories, all percentages relative to 223 papers

CFE = "Case Study, Field Study or Empirical Study", **LS** = "Literature Study", **MT** = "Model or Taxonomy", **PM** = "Processes and Methods", **RAT** = "Reference Architecture and Tools"

Investigating used methods and contributions per thematic category (see figure 5.9), the first thing we noticed was that even while "Evolution Management" contained most processes and methods overall (66,1%), every paper in "Model-driven Approaches" contained a process or a method. Also, most case studies, field studies or empirical studies relative to a category's size were conducted in "Antipatterns and Bad Smells" (85,7%) however, the absolute maximum amount of such studies were conducted in "Maintainability Metrics and Prediction" (38). Furthermore, also relative to the size of the category, most literature studies were conducted in "Service Identification and Decomposition" (28,6% of all papers in this category) but when looking at absolute numbers, most literature studies can be found in "Maintainability Metrics and Prediction" (11 papers). Moreover, the highest absolute number of models or taxonomies can be found in "Evolution Management", while "Model-driven Approaches" contains the highest amount (66,7%) relative to this category's size. Closing, "Maintainability Metrics and Prediction" contains the overall most models or taxonomies (38) but regarding the distribution inside each category, "Antipatterns and Bad Smells" contains the highest relative amount (85,7%).

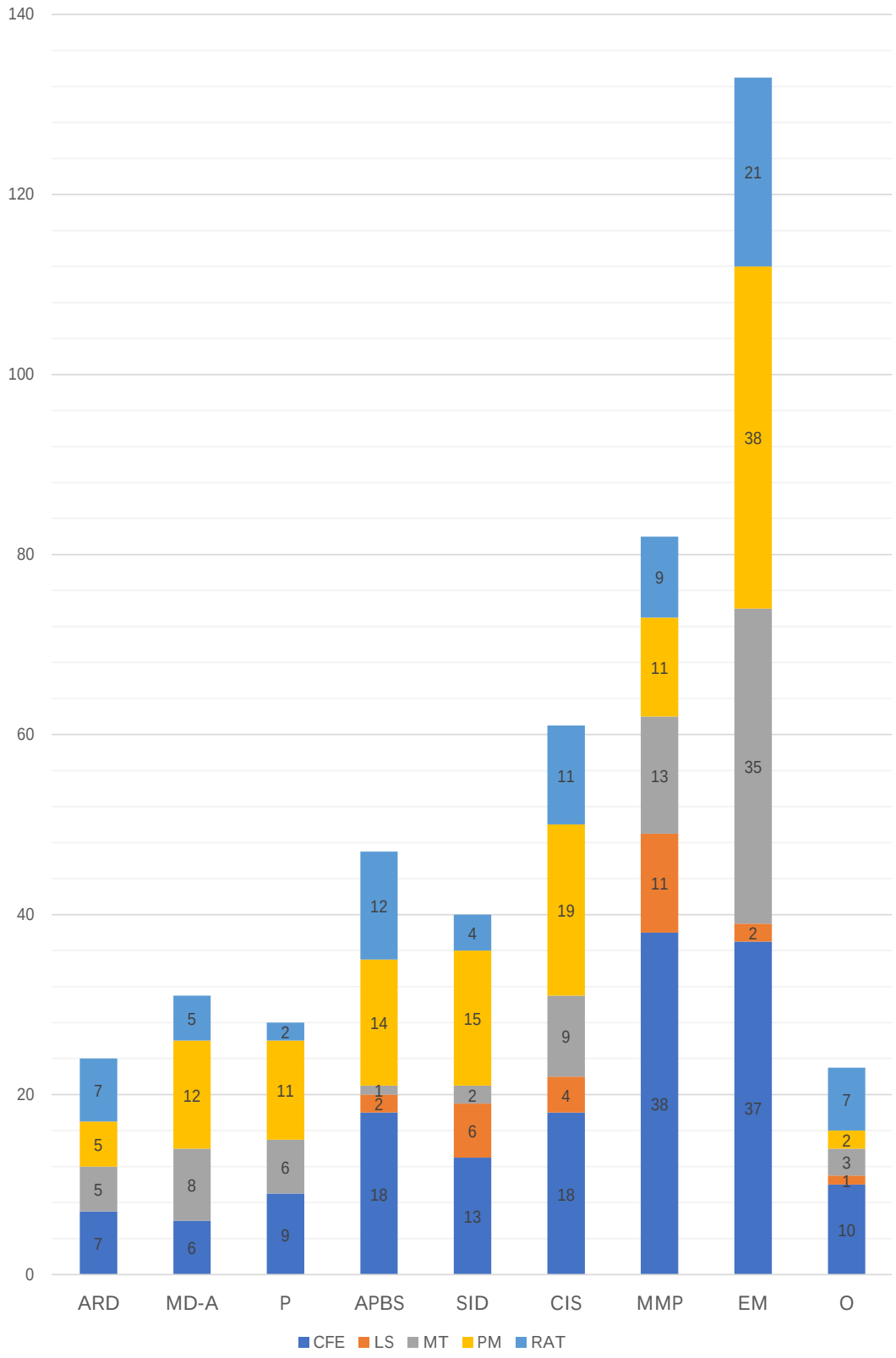

Figure 5.9: Thematical categories and therein presented approaches and contributions, multiple categories per paper are possible

ARD = “Architecture Recovery and Documentation”, **MD-A** = “Model-driven Approaches”, **P** = “Patterns”, **APBS** = “Antipatterns and Bad Smells”, **SID** = “Service Identification and Decomposition”, **CIS** = “Change Impact and Scenarios”, **MMP** = “Maintainability Metrics and Prediction”, **EM** = “Evolution Management”, **O** = “Other”, **CFE** = “Case Study, Field Study or Empirical Study”, **LS** = “Literature Study”, **MT** = “Model or Taxonomy”, **PM** = “Processes and Methods”, **RAT** = “Reference Architecture and Tools”

5.4 Relevant Research Directions, Challenges and Gaps (RQ3)

After we presented our categorization approach and the distribution of all papers among those formed categories, we now take a look at the therein presented relevant approaches and techniques. A full list with all categorized papers can be found in appendix A and appendix B.

Architecture Recovery and Documentation

Out of all nine papers retrieved in this category, seven include a case study, field study or empirical study.

Concerning SOA, three papers provide information on how a specific service-based system works or is constructed in order to ease maintenance in such systems ([BWH18], [REW+11] and [EZG11]). In an enterprise SBS such task can not be done manually due to size and complexity. Therefore, Buchgeher et al. in corporation with Raiffeisen Software GmbH provide a platform capable to extract and provide architectural information of large software systems [BWH18].

Further, such provided architectural information can be used for conformance checking against reference architectures. Weinreich et al. provide a fully automatic approach to extract an SBS architecture which is then validated against a reference architecture [WB14b].

Moving to microservices related papers, all papers provided contain preliminary work ([AAE18] and [SKH+17]) or concrete approaches ([MW18], [KUSM18] and [GCF+17]) to recover microservice-based architectures. One of these approaches is MICROLYZE which combines static and dynamic runtime data to reconstruct microservice-based architectures and recognizes infrastructural changes in real-time [KUSM18].

Model-driven Approaches

A first interesting finding we had was that all papers either contained a process or a method and no literature studies were conducted in this category.

Regarding a service’s underlying business process, we could find six papers ([BABM11], [SS09], [K C13], [Lar08], [LCZ12] and [DCG11]), which refer to Business Process Model and Notation (BPMN) or Business Process Execution Language (BPEL). A unique approach from these papers we retrieved was provided by Boukhebouze et al.. In their approach a business process is specified with rules based on the event-condition-action model. These rules can be translated into a graph of rules. The graph is then used to assess a business process’s flexibility and to estimate its cost of change [BABM11].

Another, rather unique method was presented by Liu et al. To simulate a service-oriented system's evolution process, petri nets are a potentially usable formal method. However, it is a difficult task to model the dynamic interaction behaviour of SBS because petri nets are to limited to deal with complex message structures in service interaction [LCZ12]. To solve this problem, Liu et al. propose the usage of colored, reflective petri nets, since they are more expressive compared to traditional place/transition petri nets [LCZ12].

Two of the papers which do not refer to business process related topics, make usage of patterns to improve maintainability or evolvability in SOA ([ZDZ10] and [TSZ18]). An example for the usage of patterns can be seen in [TSZ18]. In their paper, Tragatschnig et al. propose a change pattern based approach in order to support event-driven-service-oriented-architecture (EDSOA) evolution [TSZ18].

Lastly, we found out that the majority of papers in this category were released in 2013 and earlier, with only two exceptions: 1) [ZGZW16] in 2016 and 2) [TSZ18] in 2018.

Patterns

Patterns usually provide reusable templates which can help to build well defined services [WHHC14] and are tightly coupled to characteristics like maintainability. We discovered that the representation is very low for SOA and even lower for microservice-based architectures ([PDMG14], [ZNL17] and [Bog18]). In total, we found 14 papers related to patterns. With further investigating these provided papers, we found several sub-groups and few unique papers which we will describe now.

In [Ath17] and [WHHC14] the authors describe, inter alia, new patterns. Coherent with this discovery, we also found out that currently there is no paper which provides a general overview on patterns in SOA and microservices. All papers we retrieved focused either on few patterns or on a single pattern which is described further.

In total, three papers were related to pattern detection ([MPS14], [PDMG14] and [DMT13]). The authors in this papers presented two main approaches (rule-based in [DMT13] or heuristic-based in [PDMG14] and [MPS14]) to detect patterns in SOA and in case of [PDMG14] in both, SOA and microservices.

The only microservices exclusive approach, is provided by Zdun et al.. While decomposing a legacy system into a microservice-based system it is important to assess if the proposed system design is still in conformance with the planned architecture. In their work, Zdun et al. provide a set of constraints and metrics to check pattern conformance in a microservices decomposition architecture [ZNL17].

Another unique contribution, exclusive for SOA, is provided by Palma et al. In their paper, they present the results of an empirical study, which aimed to quantify the impact of service patterns as well as antipatterns on the maintenance of SBS [PAK+14].

Lastly, we found several papers providing assistance in evolution-related issues (e.g [Xa12], [PMH+15] or [Bog18]). For example, in [Xa12], the authors provide an approach to support dynamic evolution of SOA in clouds, basing on a set of several evolution patterns.

Antipatterns and Bad Smells

In contrast to patterns which help improving code by introducing good practices, antipatterns (if practiced) do exactly the opposite. Antipatterns are known for worsening software maintainability. Our first impression investigating this category was that antipatterns (21 contributions) have had slightly more attention over the last years in comparison to patterns (14 contributions).

When taking a look at the distribution between SOA and microservices related papers, we found only two papers ([PGM+15], [PDMG14]) contribute to microservices and SOA related topics as well as two papers ([CBD18], [TL18]) that are microservices exclusive. We could also find two literature studies, both related to SOA ([SPR+19], [Pal15]). One of these SLRs which was conducted by Sabir et al. focused on smell detection in Object Oriented Programming (OOP) and SOA ([SPR+19]).

Another interesting finding we had in this section was that nearly all papers contained a case study, field study or empirical study, with exception of three papers ([PM15], [SPR+19], [NPMG13]).

Concerning SOA exclusive approaches, most papers (12) we retrieved, focused on the detection of antipatterns with several approaches. One of these approaches is SODA (Service Oriented Detection for Antipatterns). SODA is a heuristic approach to detect antipatterns in SBS, provided by Nayrolles et al. in [NPMG13]. This approach gets further improved in [NMV13] by Nayrolles et al.. SOMAD (Service Oriented Mining for Antipattern Detection) improves the detection of SOA antipatterns by mining execution traces [NMV13].

A unique approach we retrieved was provided by Pismag in [Pis17]. In his work he proposes a method using machine learning to predict web service antipatterns for SOA.

Moving to contributions which are related to microservices and SOA, we found, as already described, two papers. We already mentioned the first paper ([PDMG14]) in the patterns section. In the second paper, Palma et al. present the DOLAR (Detection Of Linguistic Antipatterns in REST) approach, which uses semantic as well as syntactic analyses to detect linguistic antipatterns in RESTful Application Programming Interfaces (APIs) [PGM+15].

Regarding microservices exclusive papers, we could find only two contributions. The first contains a basic overview of bad smells in microservice-based architectures ([TL18]) and the second paper presents nine pitfalls (bad smells) when migrating from legacy architecture to microservices ([CBD18]).

Service Identification and Decomposition

A total of 21 papers were sorted into this category. From these 21 papers two are exclusively related to microservices ([ZNL17] and [JTKB18]). Both papers provide a SD approach. The most chosen type of contribution was either a process or a method (15 papers) as can be seen in figure 5.9.

We found out that literature studies were published quite regularly concerning SI strategies or challenges in SOA (e.g [KKCR09b] in 2009, [CLW11] in 2012, [HPD+14] in 2014 and [BB18a] in 2018). We briefly summarized two papers as an example to present an impression of these provided SLRs,

In their work, Cai et al. provide knowledge on when and how service identification should be carried out in SOA. They analyzed recent research and extracted sets of activities which are shared through different identification strategies. These so called high value activities were analyzed with regard to their input, artifacts and their mode of operation [CLW11].

In the process of service identification in SOA, challenges can arise. To get an overview of these challenges, Bani-Ismail et al. conducted an SLR on service identification challenges [BB18a]. They extracted eight service identification challenges namely: service quality attributes, business-IT alignment, systematic SIM, comprehensive SIM, tool support, validation, input artifact and lastly configurability of SIM. Bani-Ismail et al. also found out that service quality attributes needs further attention, since this is considered a top challenge [BB18a].

Investigating into relevant research directions, we found out that we could identify three topics, which recur more often than the rest. As this category is not well represented overall (21 papers, as already mentioned), our findings might not be representative.

The first research direction we could identify, provides knowledge on how SI strategies could be used to provide reusable services ([LMN10], [HLK08] and [AS16]).

Another research direction we found, contains papers were authors use SI techniques to automatically derive services ([LM12] and [LPM15]).

In [OSIS16], [KD08] and [EKM07], the authors focus not only on SI but also on SD as a strategy in SI.

The last direction we could find focuses only on SD strategies ([Koh08], [AZM+15] and [DOK+17]). We could not investigate if the authors perceived SD as part of SI, since there were no indications given in their paper.

One interesting paper we found contained an approach to decompose a service without access to the actual implementation. In order to accomplish such task, Athanasopoulos et al. proposed an approach to enable cohesion-driven decomposition of service interfaces. Their approach does not rely on knowledge about a service's implementation and splits a services interface into more cohesive interfaces. They validated their approach on 22 services, provided by Amazon and Yahoo [AZM+15].

Yousef provided another unique approach. In his work he presents a framework to assess the quality of a service identified with SI approaches. The Framework is demonstrated by using a case study from the health care domain [You16].

Concerning microservices, we retrieved only two papers ([ZNL17] and [JTKB18]) as already mentioned above. Both paper focus on SD in a microservice-based environment. We already presented one paper ([ZNL17]) in the pattern category. Another approach, is presented by Josélyne et al. in [JTKB18]. In their paper, they provide a method to partition services in a microservice-based system, following the domain driven design approach.

Change Impact and Scenarios

Maintaining software components in SOA and microservice-based systems, one important thing to be done is to examine the impact a change could have on other components. Analyzing this category, we found out that nearly half of all 39 papers presented therein, contain either a process or a method (19 papers). Furthermore we found out that only four literature studies were conducted ([WW13] in 2013, [AAA14] in 2014, [AAA+15] in 2015 and [SJ18] in 2016). Concerning microservices, we found a total of five papers ([SAM15], [WKZ14], [Bog18], [EZG15] and [EZG14]) which are related to microservices as well as SOA. To our knowledge there are currently no existing microservices exclusive papers.

Investigating the content presented in all papers in this category, we could find three relevant research directions: 1) change impact in business processes (e.g. [BABM11] and [WYZ10]), 2) change impact in services (e.g. [RP12] and [WKO16]) and 3) change impact prevention (e.g. [CTA14] and [BMSZ09]).

We will now briefly present an approach from each research direction in the following three paragraphs.

Business processes need to be flexible in order to be able to easily accommodate changes [BABM11]. Boukhebouze et al. provide an approach to evaluate how flexible a specific business process is. In their approach, a business process is specified with rules based on the event-condition-action model. These rules can be translated into a graph of rules. The graph is then used to assess a business process flexibility and to estimate its cost of change [BABM11].

It is important that developers are not only aware of possible dependencies in SOA but also where and if they will happen. Traditional approaches often rely on dependency graphs between web services [Dam14]. To predict change impact in a web service ecosystem Dam proposes a different approach through mining the version history [Dam14] by also assuming that if services were frequently changed together in the past, they will likely be changed together in the future.

An important activity while evolving a service is to ensure its backwards compatibility. To address this problem, Borovskiy et al. suggest a new interface design, which is called Generic Web Services. This interface design allows a service provider to add new functionality and to ensure the compatibility with already existing clients [BMSZ09].

Moving to papers related to microservices and SOA, we mostly found investigative work such as interviews ([WKZ14], [EZG15] and [EZG14]). For example, in [EZG15] Espinha et al. provide an interview conducted with six developers to learn their experiences in connection with evolving web APIs. Also among other topics in their work, they provide a study of evolution policies from API providers (Google Maps, Facebook, Netflix and Twitter) and a list of nine recommendations for API users as well as their providers [EZG15].

There were also some publications related to more general scenario-based evaluation. One example of these is [LRRA11]. In their paper, Leotta et al. describe their first step of an ongoing project. They compare two alternative architectures (non-SOA and SOA), of a postal system using SAAM (Software Architecture Analysis Method). First results have shown that SOA is capable of adapting faster to change than the old architecture. According to Leotta et al., SAAM is easy to apply and use, but the results are partially subjective [LRRA11].

Maintainability Metrics and Prediction

Containing a total of 55 papers, “Maintainability Metrics and Prediction” is the second largest category. More than half of all provided papers contain a case study, field study or empirical study. Further, this category contains the greatest absolute amount of literature studies (11 papers), as can be seen in figure 5.9. Overall, 4 papers in this category are related to SOA and microservices ([BWZ17a], [BFWZ18], [BWZ17b] and [Bog18]) while only one paper is exclusively related to microservices ([ELBH18]).

We could identify several research directions related to maintainability metrics or maintainability prediction. The first research direction contains papers where authors either find or present metrics related to structural or quality attributes (e.g. [QX09] in SOA or [BWZ17a] for SOA and microservices). To provide a short overview of such papers, we briefly summarized three contributions below.

Meaningful metrics are needed for calculating a system’s complexity. To fix this state, Qingqing et al. present a set of metrics which are usable to asses complexity in SBS [QX09].

In order to provide metrics, which can measure the degree of cohesion in SBS, Perepletchikov et al. redefined notions of cohesion metrics for OOP and procedural design in order to be applicable to SOA [PRF07].

High coupling influences maintainability negatively. It is important to quantify this design attribute with metrics. Perepletchikov et al. propose a set of metrics suitable to measure structural coupling in SOA which can be used to predict maintainability [PRFT07].

Another research direction we found contained papers where the authors provided approaches or methods to evaluate a system’s component maintainability or related quality attributes (evolvability or adaptability), as in [SP15] (SOA) or [ELBH18] (microservices).

In their paper, Senivongse et al. present a model, able to determine a service-oriented system’s maintainability [SP15]. The model uses the Quality Model of Object Oriented Design (QMOOD) assessment method, which is able to define relations between quality attributes and design properties [GJ14]. The presented assessment model can also be modified by organizations and assess which parts of service operation management or service design should be improved [SP15].

In order to provide maintainable microservice-based systems, it is important to assess if the current architecture incorporates bad practices which could lead to negative attributes such as high coupling. To support the evaluation of a microservice-based system’s architecture, Engel et al. propose MAAT (Microservice Architecture Analysis Tool). This tool is able to automatically evaluate a system’s architecture and supports the detection of hot spots therein [ELBH18].

The last research direction we could identify, contains papers where authors focused on predicting system quality attributes (evolvability, maintainability etc.), as in [WKO16] (SOA) or [KRS17] (SOA). To provide an example for this research direction, we briefly summarized an approach provided by Wang et al. in [WKO16]. This paper gives a good idea what benefits prediction-based approaches can provide.

When service interfaces evolve, service subscribers can be affected in various intensities. An unusable service due to interface incompatibilities needs to be maintained and is therefore not available for a certain amount of time. If interface evolution could be predicted, companies could manage

resources (e.g. programmers) as well as maintenance more efficiently [WKO16] to accommodate possible changes. In order to enable the prediction of a web service's evolution, Wang et al. propose an approach using machine learning based on artificial neural networks [WKO16]. In order to validate their results, they conducted experiments with six prevalent web services. The retrieved results from those experiments show that their approach is capable of predicting the evolution of a web service with a precision and recall of more than 82% [WKO16].

Evolution Management

We sorted all papers in this category which help to plan or improve software evolution and are not suitable for our other categories (see also section 5.2). Therefore we could not find any relevant research directions, otherwise a separate category would have been created. However, before we present some approaches from this category, we will briefly go over some numbers regarding our methodical categorization.

As already shown in figure 5.9, most dominant methodical categories are "Case Study, Field Study or Empirical Study", "Model or Taxonomy" and "Processes and Methods". About 66% of all papers in this category contain at least one of the mentioned methodics/ contributions. Only two papers include a literature study ([WW13] and [FL09]).

Concerning microservices, we could not find any papers exclusively related to microservices. Three papers are related to SOA and microservices ([XYCL17], [EZG15] and [EZG14]).

To get an overview which papers were sorted into this category, we present a brief summary of five papers in the following paragraphs.

One reason which can cause evolution is a change in requirements. This so called requirement-driven evolution can happen for several reasons. Such reasons can be stakeholders, who want to weaken or strengthen their requirements [SLM12], or changes in an environment. In order to support this type of software evolution, Zhang et al. present an RGPS (Role, Goal, Process and Service)-based requirement evolution framework. Even if their work is still at a formative stage, they included ideas which could help to further understand how to manage requirement evolution [ZYL11].

A problem with SOA is that components are often distributed and not contained in a single entity's domain [Fok14]. In order to support evolution of service systems, Fokaefs presents in his thesis WSDarwin, which is a framework to support evolution. WSDarwin's contribution is threefold: Firstly, it contains an Eclipse plugin which is capable of automating the adaption of service clients in Java towards web services. Secondly, it provides a web application which invokes the WSDarwin functionality which generates Web Application Description Language (WADL) interfaces and can compare different versions of these interfaces. The results are presented to the user through a JavaScript interface. Lastly, WSDarwin provides a system which can support decisionmaking in an ecosystem of competitive providers and clients [Fok14].

Development and maintenance of SOA systems require different types of staff compared to traditional systems [KLS07]. In their work, Kajko-Mattsson et al. provide a framework of required IT roles for SOA development and maintenance. Neither is the framework complete nor are the therein presented roles validated [KLS07]. However, this framework could give a first impression of required roles and their tasks.

Manual coding is a time-consuming and error-prone task. In order to reduce manual coding effort and reconfiguration when web services in SOA evolve, Zuo et al. present a programming framework which is able to support web service development, service- as well as client-side [ZYA14].

Web APIs are integrated in several service- and microservice-based systems. When these APIs evolve, this usually affects more than one party (e.g. Google Maps API). Espinha et al. conducted an interview with six web API client developers to learn from their experience with web APIs that evolved. Further they carried out a study on evolution policies from four web APIs (Google Maps, Facebook, Twitter and Netflix). They also investigated ten open source clients which use these APIs and examined their impact on the source code of these clients. Lastly, they provide a list of seven recommendations for developers of web APIs as well as client applications integrating a web API [EZG14].

Other

As already described, this category contains all papers we could not sort into one of the categories above. To provide an impression what was sorted into this category, we present two papers as an example.

Depending on service type, reusability can be a desired attribute (see also section 2.3.2). In their work, Dan et al. provide four key challenges in service reuse: 1) governing enterprise-wide use of consistent business terms, 2) governing new service creation and discovery of existing services, 3) governing service entitlement and 4) governing service enhancement. In order to address these challenges, Dan et al. recommend pro-active measures such as methodologies or supporting infrastructure capabilities [DJC08].

A main concern in SOA after deployment, is maintenance and evolution [LS08]. In order to support this activity, Lewis et al. presents an overview on SOA concepts, best practices for implementation and most important, research challenges for SOA maintenance and evolution. Four main research challenges are mentioned in their paper: 1) tools as well as techniques and environments to support maintenance activities in SOA, 2) multilanguage system analysis and maintenance, 3) re-engineering processes for migration to SOA environments and 4) evolution patterns of service-oriented systems [LS08].

5.5 Notable Differences between Service-Oriented Architecture and Microservices (RQ4)

Comparing the distribution of microservices related papers among all thematical categories, the first thing notable is that “Architecture Recovery and Documentation” not only contains most microservices exclusive approaches but also more approaches related to microservices (5) than approaches related to SOA (4). Secondly, containing an overall amount of 12 papers, “Model-driven Approach” is the only category including papers only related to SOA. Thirdly, while containing the overall most amount of papers, “Evolution Management” contains only 12,5% of all microservices related and microservices exclusive papers. In summary, as in most categories microservices related approaches are vastly outnumbered by SOA related approaches, which also can be seen in figure 5.10.

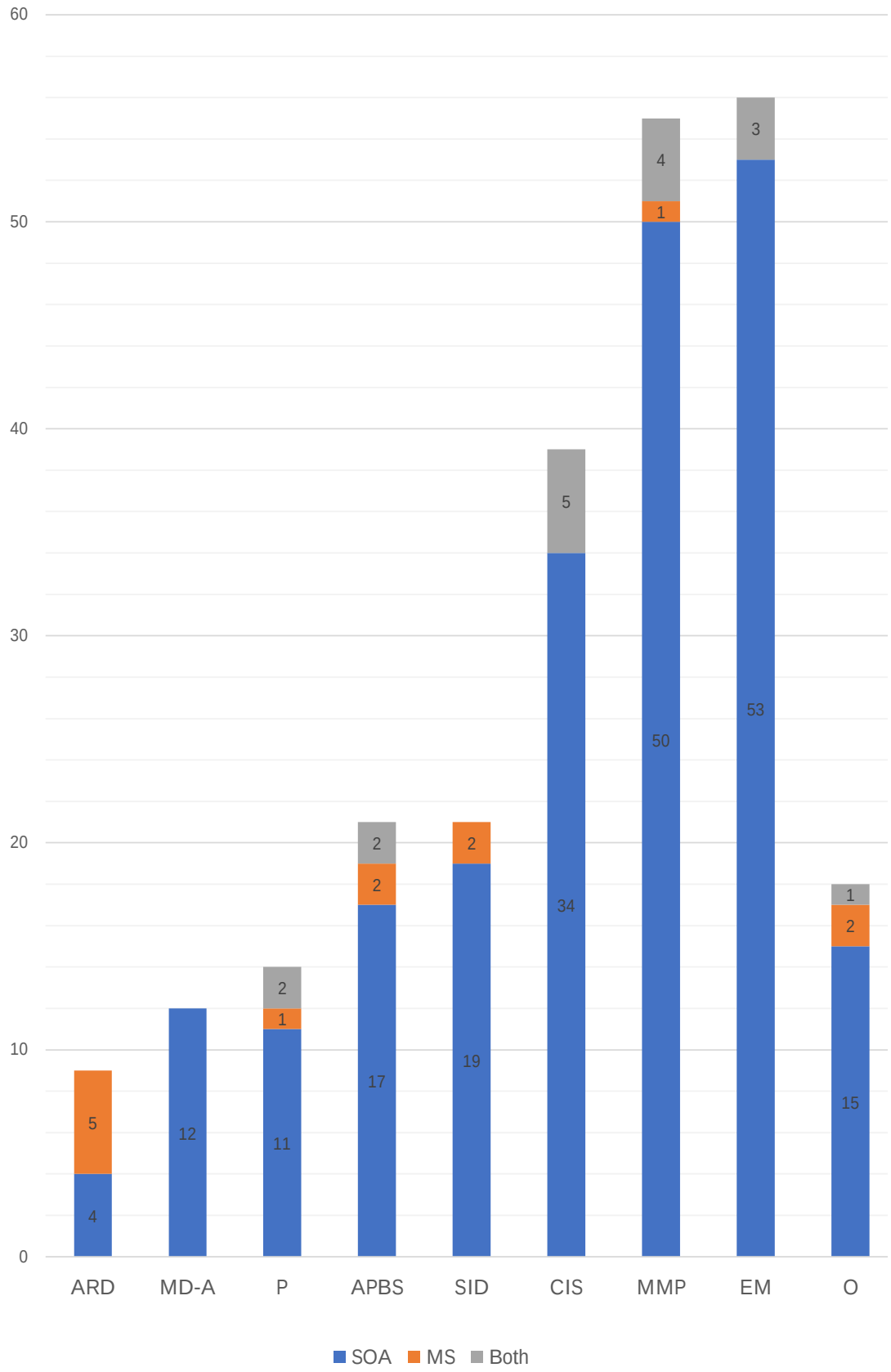

Figure 5.10: Distribution among all thematical categories

SOA = "Service-Oriented Architecture", **MS** = "Microservices", **ARD** = "Architecture Recovery and Documentation", **MD-A** = "Model-driven Approach", **P** = "Patterns", **APBS** = "Antipatterns and Bad Smells", **SID** = "Service Identification and Decomposition", **CIS** = "Change Impact and Scenarios", **MMP** = "Maintainability Metrics and Prediction", **EM** = "Evolution Management", **O** = "Other"

With figure 5.10, we have shown already the most prominent difference that is the overall amount of microservices related papers. Therefore we will not mention this difference again in the following sections. However, while going through our formed categories, we could find some more differences. We present those differences separated per category. A category will not be mentioned in this section if no notable difference could be found.

Architecture Recovery and Documentation

This category is the only one where distribution between SOA and microservices related papers is nearly equal. Four papers include approaches exclusively related to SOA (44%) and five papers contain approaches exclusively for microservices (56%). Being the smallest category overall, it contains the highest relative amount of microservices related papers.

Concerning microservices related approaches, we found something interesting namely, all of these approaches were reactive. All papers retrieved are focusing on how a systems architecture can be recovered ([MW18], [KUSM18] and [GCF+17]) but no paper provided a concrete approach to properly document a microservice-based architecture in the first place. However, concerning SOA related approaches, we could find papers which focus on architecture extraction (e.g [EZG11]) and documentation (e.g [BWH18]).

Model-driven Approaches

"Model-driven Approaches" is the only category, which contains no papers related to microservices, as can also be seen in figure 5.10. All papers (12) are exclusively related to SOA.

Patterns

In comparison to SOA related approaches, patterns are used in microservices related papers to also verify architecture conformance ([ZNL17]).

Another difference we could find was that in SOA related papers, authors used patterns for change impact estimation ([WHHC14]), an approach we could not find in microservices related patterns.

Antipatterns and Bad Smells

Regarding "Antipatterns and Bad Smells", we found out that papers exlusively related to microservices only provide basic information on antipatterns ([CBD18] and [TL18]), but no approaches to detect them. SOA related papers however, not only provide basic knowledge about antipatterns, but also contain different approaches (e.g. SODA-W in [PMTG14]) to detect antipatterns.

Service Identification and Decomposition

A difference we noticed in this section was while papers related to SOA provided content on service identification and service decomposition (e.g. [CLW11]), both microservices exclusive papers only included content related to service decomposition ([ZNL17] and [JTKB18]).

6 Threats to Validity

In this section, we will explain possible threats to the validity of this SLR and our approach to counter or mitigate these. We extracted possible threats (in addition to our own) from literature (Zhou et al. [ZJZ+16]), and provide our approach to counter or mitigate these threats (if possible).

6.1 Planning Phase

In this phase, we present all possible threats which could endanger the validity of this thesis during the planning of the SLR.

Non-specification of Details

This threat could lead to non-repeatability of this SLR. In order to counter this threat, we developed a rigid research protocol (as can be seen in chapter 4), including all necessary details.

Insufficient Search Strategy

An insufficient search strategy could return undesirable research results and therefore not return the type of papers this thesis intended to explore. To counter this risk, we provide a protocol (chapter 4) which was developed and then reviewed by another researcher.

Lack of Standardization

A lack of standardization could lead future researchers into misunderstanding findings we presented in this work. To counter this threat, we provide an overview on important technical backgrounds and terms in chapter 2.

Incomprehensive Databases

Choosing the wrong databases could lead to receiving papers not related to the desired context (i.e. computer science). To counter this problem, we chose databases related to software engineering or restricted the search term further as can be seen in section 4.2.1.

Inappropriate Inclusion and Exclusion Criteria

Inappropriate inclusion or exclusion criteria could not only reduce reproduceability but also lead to the inclusion of low quality papers. In order to counter this threat, inclusion and exclusion criteria were developed in an iterative process and peer reviewed by a second researcher.

Inadequate Sample Size

As we already described in section 4.2.4, we chose to limit the maximum results of each database to 250 results in order to provide a manageable amount of work. This has not only limited our sample size, but also restricted our intermediate result set to results which were interpreted as relevant by the search engine. We could not provide a sufficient countermeasure to this threat. However, we conducted a forward citation search which should have weakened this threat.

6.2 Conducting Phase

In this phase, we present all possible threats which could endanger the validity of this thesis during the conduction of this SLR.

Bias in Study Selection

A bias in the selection of studies could lead to the inclusion of papers which should not be in this thesis and to the exclusion of papers which should be in it. To mitigate this threat, the inclusion and exclusion process was done independently by two researchers and then discussed.

Source not Accessible

We already saw this as a possible problem before conducting the SLR. However, we did not encounter an inaccessible source.

Categorization Bias

A possible threat to validity could be that we have not developed an optimal set of categories. In combination with “Subjective Categorization” this could lead to a non-representative set of categories and a non-representative representation of our retrieved papers. To mitigate this risk, the categories were developed by two researchers in several iterations.

Subjective Categorization

A subjective categorization could lead to the misinterpretation of our results. For example it could lead to misinterpretation while detecting possible gaps (e.g. low amount of papers related to microservices in a specific category). In order to limit this problem, the categorization process was also done independently by two researchers and different results were discussed.

Subjective Data Interpretation

The subjective interpretation of papers could lead to the wrong synthesizing of its content. In order to mitigate this threat, papers which were unclear regarding their contribution were discussed among two researchers.

6.3 Reporting Phase

In this phase, we present a possible threat which could endanger the validity of this thesis during the reporting phase.

Bias Concerning Relevant Papers

A potential risk is that other researchers may have chosen other papers as relevant approaches (regarding **RQ3**). We tried to mitigate a potential bias in pointing out more general, objective research directions and underpinned these with examples. Further, several proof readings were conducted with a more experienced researcher.

Lack of Expert Evaluation

This threat could lead to a wrong conclusion of this study. To prevent this, the study was proof read several times during the writing process by another researcher.

6.4 Non-stage Specific Threats

Failures while conducting a study can happen in several stages. Possible reasons were listed above. A further possible threat, which does not go well with one of the mentioned stages, is the inexperience of the researcher. But as with nearly all listed possible threats we tried to encounter this problem with various proof readings by another, more experienced researcher.

7 Conclusion

In this chapter, we present the summary of this thesis, our implications and an outlook on possible future work.

7.1 Summary

In chapter 1, we began this paper providing a motivation on why there is a need to conduct an SLR on maintainability assurance approaches and techniques for SOA and microservices. We explained that software maintenance costs money and therefore effective and efficient maintenance is important. Further we introduced SOA and microservices as promising architectural styles concerning maintainability. Moreover, even if these architectures promise to be maintainable, it is crucial to know how maintenance should be carried out. We explained that research is spread across several databases and that an SLR is capable of collecting and synthesizing this research.

Following this, we presented our four research questions:

1. **RQ1**: How can maintainability assurance approaches and techniques for service- and microservice-based systems, that have been proposed in scientific literature, be categorized?
2. **RQ2**: How are the identified publications distributed among the formed categories?
3. **RQ3**: What are the most relevant research directions, challenges or gaps per identified category?
4. **RQ4**: Are there notable differences between the approaches and techniques proposed for service-based systems and those for microservices?

After the provision of our objective, we explained some important technical background details which were important to fully understand this thesis. We started with explaining the basics of service- and microservice-based systems. Next, we described the term maintainability. We started off with a definition and different types of software maintenance and then moved forward to maintainability and influencing factors.

Directly following this chapter we provided a brief overview of related work. We presented several literature reviews, we retrieved through our research, which either provided a much broader scope (e.g. [STH18]), or a much narrower scope (e.g. [BWZ17a]). The fact that there were no papers with a similar scope to ours indicated a gap in this field of research and therefore legitimated this thesis.

Having pointed out this gap we went forward and explained how the SLR was conducted. We first provided an overview of what defines an SLR and then presented our research process in detail. We started with mentioning our research questions and then moved on to our datasources and our search strategy. Subsequently, we detailed our inclusion and exclusion process and explained how we conducted our follow-up citation search and the categorization of all papers.

Concerning the results of this study, we first presented a more general view on retrieved results. We showed how our results were distributed among the chosen publishers before and after citation search. Hereby we discovered that microservices related approaches are vastly outnumbered by SOA related approaches.

We then provided an overview of our threefold categorization approach and briefly explained each category **(RQ1)**. After this, we displayed how the papers are distributed among those formed categories and discovered a notable underrepresentation of microservices related research **(RQ2)**. Lastly, we presented relevant research directions **(RQ3)** and provided an overview on notable differences between SOA and microservice-based architecture **(RQ4)**.

Regarding **RQ1**, we provided as already described, a threefold categorization system. Each paper from our result set (223 papers) was categorized into an architectural category, a thematical category and if possible into an optional methodical category.

In **RQ2**, we analyzed the distribution of the papers among the formed categories. We discovered that microservices related approaches are vastly outnumbered by approaches related to SOA. The overall best represented category was “Evolution Management” (25,1% of all papers), shortly followed by “Maintainability Metrics and Prediction” (24,7% of all papers). However, “Architecture Recovery” contained the least amount of papers (4% of all papers).

Concerning **RQ3**, we presented our overview of relevant research directions, challenges and gaps. For example regarding SOA related papers in the category “Change Impact and Scenarios”, we could identify three relevant research directions: 1) change impact in business processes, 2) change impact in services and 3) change impact prevention. Another example are research directions in “Maintainability Metrics and Prediction”. We could coarsely divide papers on the following research directions: 1) finding a provisioning of metrics, 2) using metrics to evaluate a system’s maintainability (or a related quality attribute) and 3) prediction of a system’s quality attribute with metrics. We also found papers which could not be assigned to a relevant or frequent research direction. A lot of these papers can be found in “Evolution Management” and “Other” due to their generalization.

In **RQ4**, we presented notable differences between SOA and microservice-based architecture. For example, we noticed that even if microservices related approaches occurred rarely among all categories, we identified one category (“Architecture Recovery and Documentation”) where SOA related papers (4) were barely outnumbered by microservices related papers (5). Another interesting finding we had, for example was that most approaches of microservices related papers in “Architecture Recovery and Documentation” were reactive (only focusing on architecture recovery). A last example for such notable differences can be seen in “Model-driven Approaches”. We could not retrieve any microservices related paper in this category. However, it is difficult to derive if this constitutes a research gap or if model-driven approaches are not suitable to assure maintainability in microservice-based systems. As can be seen in [TDKL18], papers related to model-driven approaches do exist for microservices, but we did not find any during our study. A possible reason for this might be that they do not cover maintainability/evolvability related aspects.

7.2 Implications and Future Work

In this paper, we discovered several things. Firstly, research regarding maintainability assurance in microservice-based systems has not gotten a lot of attention over the last years (see also figure 5.6). A possible reason for this state could be that microservices are seen not as important as SOA and therefore get less attention. Another imaginable reason is that microservice-based systems are perceived to have a positive impact on a software product's maintainability [BFWZ19], without further attention. Furthermore, more than half of all retrieved papers were released before 2014. There is a good chance that papers released before 2014 may have lost relevance to some degree. Lastly, we could identify several differences between approaches suitable for SOA or microservices. Based on this take-aways we suggest the following areas for further research:

Firstly, since we have shown a massive underrepresentation of microservices related maintainability assurance research, follow up research should fill this gap. In order to achieve this, we suggest that future researchers should first investigate in which area of maintainability assurance urgent need exists. Our categorization can hereby provide a first idea of where to start.

Secondly, even if maintainability assurance in SOA was well represented in this study, it might be a good idea to revisit these areas of research. This is because of the possibility that requirements and use cases of SOA could have changed over the years due to technological advancement.

Thirdly, we presented a paper ([BWZ17a]) where the authors investigated if existing SOA related maintainability metrics are also suitable for microservices. A possible idea for further research we derived from this paper, could therefore be to investigate if approaches exist throughout our result set, which are also applicable to microservices or the other way around.

Fourthly, during our research, we could not retrieve any model-driven approaches specifically designed for microservices. As we have shown in section 7.1, papers related to model-driven approaches do exist for microservices [TDKL18]. Further research could investigate if existing model-driven approaches also aim to improve the maintainability of microservice-based systems or if there is no interest in this topic.

Lastly, as we mentioned in section 4.2.4, we excluded runtime adaptation, testing and legacy migration since each of these topics could provide enough content to be dealt separately with. We therefore suggest future researchers to pick up these topics in a literature study.

Bibliography

[17] "ISO/IEC/IEEE International Standard - Systems and software engineering–Vocabulary". In: *ISO/IEC/IEEE 24765:2017(E)* (Aug. 2017), pp. 1–541. DOI: 10.1109/IEEESTD.2017.8016712 (cit. on pp. 22, 24).

[AAA+15] K. A. Alam, R. Ahmad, A. Akhunzada, M. H. N. M. Nasir, S. U. Khan. "Impact analysis and change propagation in service-oriented enterprises: A systematic review". In: *Information Systems* 54 (2015), pp. 43–73. ISSN: 0306-4379. DOI: https://doi.org/10.1016/j.is.2015.06.003. URL: http://www.sciencedirect.com/science/article/pii/S0306437915001179 (cit. on pp. 49, 93, 95).

[AAA14] K. Amjad Alam, R. B. Ahmad, M. Akhtar. "Change Impact analysis and propagation in service based business process management systems preliminary results from a systematic review". In: *2014 8th. Malaysian Software Engineering Conference (MySEC)*. Sept. 2014, pp. 7–12. DOI: 10.1109/MySec.2014.6985981 (cit. on pp. 49, 93, 95).

[AAE18] N. Alshuqayran, N. Ali, R. Evans. "Towards Micro Service Architecture Recovery: An Empirical Study". In: *2018 IEEE International Conference on Software Architecture (ICSA)*. Apr. 2018, pp. 47–4709. DOI: 10.1109/ICSA.2018.00014 (cit. on pp. 45, 93, 95, 96).

[ACMT18] K. Alkharabsheh, Y. Crespo, E. Manso, J. A. Taboada. "Software Design Smell Detection: a systematic mapping study". In: *Software Quality Journal* (Oct. 2018). ISSN: 1573-1367. DOI: 10.1007/s11219-018-9424-8. URL: https://doi.org/10.1007/s11219-018-9424-8 (cit. on p. 26).

[AD10] J. L. Arciniegas H., J. C. Dueñas L. "Evolvability Characterization in the Context of SOA". In: *Advances in Software Engineering*. Ed. by T.-h. Kim, H.-K. Kim, M. K. Khan, A. Kiumi, W.-c. Fang, D. Ślęzak. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 242–253. ISBN: 978-3-642-17578-7 (cit. on pp. 94–96).

[AGR13] D. Allam, H. Grall, J.-C. Royer. "The Substitution Principle in an Object-Oriented Framework for Web Services: From Failure to Success". In: *Proceedings of International Conference on Information Integration and Web-based Applications & Services*. IIWAS '13. Vienna, Austria: ACM, 2013, 250:250–250:259. ISBN: 978-1-4503-2113-6. DOI: 10.1145/2539150.2539192. URL: http://doi.acm.org/10.1145/2539150.2539192 (cit. on pp. 94–96).

[AJP12] A. Ahmad, P. Jamshidi, C. Pahl. "Pattern-driven Reuse in Architecture-centric Evolution for Service Software". In: (Jan. 2012) (cit. on pp. 93–96).

[AP10] A. Ahmad, C. Pahl. "Pattern-based customisable transformations for style-based service architecture evolution". In: *2010 International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*. Oct. 2010, pp. 371–376. DOI: 10.1109/CISIM.2010.5643631 (cit. on pp. 93–96).

[AP11] A. Ahmad, C. Pahl. "Customisable Transformation-Driven Evolution for Service Architectures". In: *2011 15th European Conference on Software Maintenance and Reengineering*. Mar. 2011, pp. 373–376. DOI: 10.1109/CSMR.2011.56 (cit. on pp. 94, 96).

[AS16] V. Alkkiomäki, K. Smolander. "Anatomy of one service-oriented architecture implementation and reasons behind low service reuse". In: *Service Oriented Computing and Applications* 10.2 (June 2016), pp. 207–220. ISSN: 1863-2394. DOI: 10.1007/s11761-015-0181-2. URL: https://doi.org/10.1007/s11761-015-0181-2 (cit. on pp. 48, 93, 95).

[Ath17] D. Athanasopoulos. "Service Decoupler: Full Dynamic Decoupling in Service Invocation". In: *Proceedings of the 22Nd European Conference on Pattern Languages of Programs*. EuroPLoP '17. Irsee, Germany: ACM, 2017, 10:1–10:9. ISBN: 978-1-4503-4848-5. DOI: 10.1145/3147704.3147716. URL: http://doi.acm.org/10.1145/3147704.3147716 (cit. on pp. 46, 93, 95, 96).

[AZ11] D. Athanasopoulos, A. V. Zarras. "Fine-Grained Metrics of Cohesion Lack for Service Interfaces". In: *2011 IEEE International Conference on Web Services*. July 2011, pp. 588–595. DOI: 10.1109/ICWS.2011.27 (cit. on pp. 94, 95).

[AZD11] S. Alahmari, E. Zaluska, D. C. De Roure. "A Metrics Framework for Evaluating SOA Service Granularity". In: *2011 IEEE International Conference on Services Computing*. July 2011, pp. 512–519. DOI: 10.1109/SCC.2011.98 (cit. on pp. 94, 95).

[AZM+15] D. Athanasopoulos, A. V. Zarras, G. Miskos, V. Issarny, P. Vassiliadis. "Cohesion-Driven Decomposition of Service Interfaces without Access to Source Code". In: *IEEE Transactions on Services Computing* 8.4 (July 2015), pp. 550–562. ISSN: 1939-1374. DOI: 10.1109/TSC.2014.2310195 (cit. on pp. 48, 93, 95, 96).

[BABM09] M. Boukhebouze, Y. Amghar, A. Benharkat, Z. Maamar. "Towards an Approach for Estimating Impact of Changes on Business Processes". In: *2009 IEEE Conference on Commerce and Enterprise Computing*. July 2009, pp. 415–422. DOI: 10.1109/CEC.2009.35 (cit. on pp. 93, 95, 96).

[BABM11] M. Boukhebouze, Y. Amghar, A.-N. Benharkat, Z. Maamar. "A rule-based approach to model and verify flexible business processes". In: *International Journal of Business Process Integration and Management* 5.4 (2011), pp. 287–307. DOI: 10.1504/IJBPIM.2011.043389. eprint: https://www.inderscienceonline.com/doi/pdf/10.1504/IJBPIM.2011.043389. URL: https://www.inderscienceonline.com/doi/abs/10.1504/IJBPIM.2011.043389 (cit. on pp. 45, 49, 93, 96).

[Bal19] Ş. Y. Balaman. "Chapter 5 - Uncertainty Issues in Biomass-Based Production Chains". In: *Decision-Making for Biomass-Based Production Chains*. Ed. by Ş. Y. Balaman. Academic Press, 2019, pp. 113–142. ISBN: 978-0-12-814278-3. DOI: https://doi.org/10.1016/B978-0-12-814278-3.00005-4. URL: http://www.sciencedirect.com/science/article/pii/B9780128142783000054 (cit. on p. 38).

[BB18a] B. Bani-Ismail, Y. Baghdadi. "A Literature Review on Service Identification Challenges in Service Oriented Architecture". In: *Knowledge Management in Organizations*. Ed. by L. Uden, B. Hadzima, I.-H. Ting. Cham: Springer International Publishing, 2018, pp. 203–214. ISBN: 978-3-319-95204-8 (cit. on pp. 26, 39, 47, 48, 93, 95).

[BB18b] B. Bani-Ismail, Y. Baghdadi. "A Survey of Existing Evaluation Frameworks for Service Identification Methods: Towards a Comprehensive Evaluation Framework". In: *Knowledge Management in Organizations*. Ed. by L. Uden, B. Hadzima, I.-H. Ting. Cham: Springer International Publishing, 2018, pp. 191–202. ISBN: 978-3-319-95204-8 (cit. on pp. 93, 95, 96).

[BBPM15] S. Balderas-Díaz, K. Benghazi, G. Prados, E. Miró. "Designing Configurable and Adaptive Systems in eHealth". In: *Proceedings of the 3rd 2015 Workshop on ICTs for Improving Patients Rehabilitation Research Techniques*. REHAB '15. Lisbon, Portugal: ACM, 2015, pp. 118–121. ISBN: 978-1-4503-3898-1. DOI: 10.1145/2838944.2838973. URL: http://doi.acm.org/10.1145/2838944.2838973 (cit. on pp. 94, 96).

[BCDP14] D. Bianchini, C. Cappiello, V. De Antonellis, B. Pernici. "Service Identification in Interorganizational Process Design". In: *IEEE Transactions on Services Computing* 7.2 (Apr. 2014), pp. 265–278. ISSN: 1939-1374. DOI: 10.1109/TSC.2013.26 (cit. on pp. 93, 95, 96).

[BCL12a] H. P. Breivold, I. Crnkovic, M. Larsson. "A systematic review of software architecture evolution research". In: *Information and Software Technology* 54.1 (2012), pp. 16–40. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2011.06.002. URL: http://www.sciencedirect.com/science/article/pii/S0950584911001376 (cit. on p. 25).

[BCL12b] H. P. Breivold, I. Crnkovic, M. Larsson. "Software architecture evolution through evolvability analysis". In: *Journal of Systems and Software* 85.11 (2012), pp. 2574–2592. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2012.05.085. URL: http://www.sciencedirect.com/science/article/pii/S016412121200163X (cit. on p. 23).

[BD13] D. V. Babu, M. P. Darsi. "A Survey on Service Oriented Architecture and Metrics to Measure Coupling". In: *International Journal on Computer Science and Engineering* 5.8 (2013), p. 726. URL: http://www.enggjournals.com/ijcse/doc/IJCSE13-05-08-058.pdf (cit. on pp. 94, 95).

[BFH16] H. Bloch, A. Fay, M. Hoernicke. "Analysis of service-oriented architecture approaches suitable for modular process automation". In: *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*. Sept. 2016, pp. 1–8. DOI: 10.1109/ETFA.2016.7733651 (cit. on p. 30).

[BFWZ18] J. Bogner, J. Fritzsch, S. Wagner, A. Zimmermann. "Limiting Technical Debt with Maintainability Assurance: An Industry Survey on Used Techniques and Differences with Service- and Microservice-based Systems". In: *Proceedings of the 2018 International Conference on Technical Debt*. TechDebt '18. Gothenburg, Sweden: ACM, 2018, pp. 125–133. ISBN: 978-1-4503-5713-5. DOI: 10.1145/3194164.3194166. URL: http://doi.acm.org/10.1145/3194164.3194166 (cit. on pp. 50, 94, 95).

[BFWZ19] J. Bogner, J. Fritzsch, S. Wagner, A. Zimmermann. "Microservices in Industry: Insights into Technologies, Characteristics, and Software Quality". In: *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. Mar. 2019, pp. 187–195. DOI: 10.1109/ICSA-C.2019.00041 (cit. on pp. 40, 63).

[BKC+10] A. Bucchiarone, R. Kazhamiakin, C. Cappiello, E. di Nitto, V. Mazza. "A Context-driven Adaptation Process for Service-based Applications". In: *Proceedings of the 2Nd International Workshop on Principles of Engineering Service-Oriented Systems*. PESOS '10. Cape Town, South Africa: ACM, 2010, pp. 50–56. ISBN: 978-1-60558-963-3. DOI: 10.1145/1808885.1808896. URL: http://doi.acm.org/10.1145/1808885.1808896 (cit. on pp. 38, 94, 96).

[BMSZ09] V. Borovskiy, J. Mueller, M.-P. Schapranow, A. Zeier. "Ensuring Service Backwards Compatibility with Generic Web Services". In: *Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*. PESOS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 95–98. ISBN: 978-1-4244-3716-0. DOI: 10.1109/PESOS.2009.5068827. URL: https://doi.org/10.1109/PESOS.2009.5068827 (cit. on pp. 49, 93, 96).

[Bog18] J. Bogner. "Towards an Evolvability Assurance Method for Service-Based Systems". In: Sept. 2018. URL: https://www.researchgate.net/publication/327690326_Towards_an_Evolvability_Assurance_Method_for_Service-Based_Systems (cit. on pp. 46, 49, 50, 93, 94, 96).

[BWH18] G. Buchgeher, R. Weinreich, H. Huber. "A Platform for the Automated Provisioning of Architecture Information for Large-Scale Service-Oriented Software Systems". In: *Software Architecture*. Ed. by C. E. Cuesta, D. Garlan, J. Pérez. Cham: Springer International Publishing, 2018, pp. 203–218. ISBN: 978-3-030-00761-4 (cit. on pp. 45, 55, 93, 96).

[BWZ17a] J. Bogner, S. Wagner, A. Zimmermann. "Automatically Measuring the Maintainability of Service- and Microservice-based Systems: A Literature Review". In: *Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*. IWSM Mensura '17. Gothenburg, Sweden: ACM, 2017, pp. 107–115. ISBN: 978-1-4503-4853-9. DOI: 10.1145/3143434.3143443. URL: http://doi.acm.org/10.1145/3143434.3143443 (cit. on pp. 23, 25, 36, 50, 61, 63, 94, 95).

[BWZ17b] J. Bogner, S. Wagner, A. Zimmermann. "Towards a Practical Maintainability Quality Model for Service-and Microservice-based Systems". In: *Proceedings of the 11th European Conference on Software Architecture: Companion Proceedings*. ECSA '17. Canterbury, United Kingdom: ACM, 2017, pp. 195–198. ISBN: 978-1-4503-5217-8. DOI: 10.1145/3129790.3129816. URL: http://doi.acm.org/10.1145/3129790.3129816 (cit. on pp. 50, 94, 96).

[CBD18] A. Carrasco, B. v. Bladel, S. Demeyer. "Migrating Towards Microservices: Migration and Architecture Smells". In: *Proceedings of the 2Nd International Workshop on Refactoring*. IWoR 2018. Montpellier, France: ACM, 2018, pp. 1–6. ISBN: 978-1-4503-5974-0. DOI: 10.1145/3242163.3242164. URL: http://doi.acm.org/10.1145/3242163.3242164 (cit. on pp. 47, 55, 93, 95).

[CCBJ11] G. Candido, A. W. Colombo, J. Barata, F. Jammes. "Service-Oriented Infrastructure to Support the Deployment of Evolvable Production Systems". In: *IEEE Transactions on Industrial Informatics* 7.4 (Nov. 2011), pp. 759–767. ISSN: 1551-3203. DOI: 10.1109/TII.2011.2166779 (cit. on pp. 94–96).

[CLW11] S. Cai, Y. Liu, X. Wang. “A Survey of Service Identification Strategies”. In: *2011 IEEE Asia-Pacific Services Computing Conference*. Dec. 2011, pp. 464–470. DOI: 10.1109/APSCC.2011.12 (cit. on pp. 47, 48, 56, 93, 95).

[CS15] A. Chomchumpol, T. Senivongse. “Stability measurement model for service-oriented systems”. In: *2015 9th Malaysian Software Engineering Conference (MySEC)*. Dec. 2015, pp. 54–59. DOI: 10.1109/MySEC.2015.7475195 (cit. on pp. 94, 96).

[CTA14] K. Chiponga, P. Tarwireyi, M. O. Adigun. “A version-based transformation proxy for service evolution”. In: *2014 IEEE 6th International Conference on Adaptive Science Technology (ICAST)*. Oct. 2014, pp. 1–5. DOI: 10.1109/ICASTECH.2014.7068123 (cit. on pp. 49, 93, 96).

[CZ11] I. Chowdhury, M. Zulkernine. “Using complexity, coupling, and cohesion metrics as early indicators of vulnerabilities”. In: *Journal of Systems Architecture* 57.3 (2011). Special Issue on Security and Dependability Assurance of Software Architectures, pp. 294–313. ISSN: 1383-7621. DOI: https://doi.org/10.1016/j.sysarc.2010.06.003. URL: http://www.sciencedirect.com/science/article/pii/S1383762110000615 (cit. on p. 23).

[Dam14] H. K. Dam. “Predicting change impact in Web service ecosystems”. In: *International Journal of Web Information Systems* 10.3 (2014), pp. 275–290. DOI: 10.1108/IJWIS-03-2014-0006. eprint: https://doi.org/10.1108/IJWIS-03-2014-0006. URL: https://doi.org/10.1108/IJWIS-03-2014-0006 (cit. on pp. 37, 49, 93, 95, 96).

[DCG11] K. Dahman, F. Charoy, C. Godart. “Towards Consistency Management for a Business-Driven Development of SOA”. In: *2011 IEEE 15th International Enterprise Distributed Object Computing Conference*. Aug. 2011, pp. 267–275. DOI: 10.1109/EDOC.2011.30 (cit. on pp. 45, 93, 95, 96).

[DD08] T. Dybå, T. Dings. “Empirical studies of agile software development: A systematic review”. In: *Information Software Technology* 50 (2008), pp. 833–859 (cit. on p. 30).

[DDD11] M. Daghaghzadeh, A. B. Dastjerdi, H. Daghaghzadeh. “A Metric for Measuring Degree of Service Cohesion in Service Oriented Designs”. In: *empty* 8.5 (2011), pp. 83–89. URL: http://www.ijcsi.org/articles/A-metric-for-measuring-degree-of-service-cohesion-in-service-oriented-designs.php (cit. on pp. 38, 94, 95).

[Dem13] B. Demchak. “Policy driven development : SOA evolvability through late binding”. PhD thesis. UC San Diego, 2013. URL: https://escholarship.org/uc/item/8d79f7r4 (cit. on pp. 94–96).

[DG10] H. K. Dam, A. Ghose. “Supporting Change Propagation in the Maintenance and Evolution of Service-Oriented Architectures”. In: *2010 Asia Pacific Software Engineering Conference*. Nov. 2010, pp. 156–165. DOI: 10.1109/APSEC.2010.27 (cit. on pp. 93, 95, 96).

[Dica] O. Dictionaries. *method | Definition of method in English by Oxford Dictionaries*. https://en.oxforddictionaries.com/definition/method. (Accessed on 05/19/2019). (Cit. on p. 39).

[Dicb] O. Dictionaries. *process | Definition of process in English by Oxford Dictionaries*. https://en.oxforddictionaries.com/definition/process. (Accessed on 05/19/2019). (Cit. on p. 39).

[Dicc] C. Dictionary. *TAXONOMY | meaning in the Cambridge English Dictionary*. https://dictionary.cambridge.org/dictionary/english/taxonomy. (Accessed on 05/19/2019). (Cit. on p. 39).

[DJC08] A. Dan, R. D. Johnson, T. Carrato. "SOA Service Reuse by Design". In: *Proceedings of the 2Nd International Workshop on Systems Development in SOA Environments*. SDSOA '08. Leipzig, Germany: ACM, 2008, pp. 25–28. ISBN: 978-1-60558-029-6. DOI: 10.1145/1370916.1370923. URL: http://doi.acm.org/10.1145/1370916.1370923 (cit. on pp. 38, 52, 94).

[DMT13] A. Demange, N. Moha, G. Tremblay. "Detection of SOA Patterns". In: *Service-Oriented Computing*. Ed. by S. Basu, C. Pautasso, L. Zhang, X. Fu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 114–130. ISBN: 978-3-642-45005-1 (cit. on pp. 31, 37, 46, 93–96).

[DOK+17] M. Daagi, A. Ouniy, M. Kessentini, M. M. Gammoudi, S. Bouktif. "Web Service Interface Decomposition Using Formal Concept Analysis". In: *2017 IEEE International Conference on Web Services (ICWS)*. June 2017, pp. 172–179. DOI: 10.1109/ICWS.2017.30 (cit. on pp. 48, 93, 95, 96).

[dVE+13] R. R. de Oliveira, R. V. Vieira Sanchez, J. C. Estrella, R. Pontin de Mattos Fortes, V. Brusamolin. "Comparative evaluation of the maintainability of RESTful and SOAP-WSDL web services". In: *2013 IEEE 7th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems*. Sept. 2013, pp. 40–49. DOI: 10.1109/MESOCA.2013.6632733 (cit. on pp. 94, 95).

[ECZG12] T. Espinha, C. Chen, A. Zaidman, H. Gross. "Maintenance Research in SOA - Towards a Standard Case Study". In: *2012 16th European Conference on Software Maintenance and Reengineering*. Mar. 2012, pp. 391–396. DOI: 10.1109/CSMR.2012.49 (cit. on pp. 94, 95).

[EKM07] A. Erradi, N. Kulkarni, P. Maheshwari. "Service Design Process for Reusable Services: Financial Services Case Study". In: *Service-Oriented Computing – ICSOC 2007*. Ed. by B. J. Krämer, K.-J. Lin, P. Narasimhan. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 606–617. ISBN: 978-3-540-74974-5 (cit. on pp. 48, 93, 95, 96).

[ELBH18] T. Engel, M. Langermeier, B. Bauer, A. Hofmann. "Evaluation of Microservice Architectures: A Metric and Tool-Based Approach". In: *Information Systems in the Big Data Era*. Ed. by J. Mendling, H. Mouratidis. Cham: Springer International Publishing, 2018, pp. 74–89. ISBN: 978-3-319-92901-9 (cit. on pp. 50, 94–96).

[EM15] A. A. M. Elhag, R. Mohamad. "Service-oriented design measurement and theoretical validation". In: *Jurnal Teknologi* 77 (Jan. 2015), pp. 1–14. DOI: 10.11113/jt.v77.6181. URL: https://jurnalteknologi.utm.my/index.php/jurnalteknologi/article/view/6181 (cit. on pp. 94, 95).

[EMK12] A. Eisfeld, D. A. McMeekin, A. P. Karduck. "Complex environment evolution: Challenges with semantic service infrastructures". In: *2012 6th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*. June 2012, pp. 1–6. DOI: 10.1109/DEST.2012.6227937 (cit. on pp. 38, 93, 95, 96).

[Erl09] T. Erl. *SOA Design Patterns*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2009. ISBN: 0136135161, 9780136135166 (cit. on p. 21).

[EW09] T. Erl, H. Wilhelmsen. *SOA Pattern (#2): Non-Agnostic Context | | InformIT*. http://www.informit.com/articles/article.aspx?p=1324282. (Accessed on 05/07/2019). Jan. 2009 (cit. on p. 22).

[EZG11] T. Espinha, A. Zaidman, H. Gross. "Understanding Service-Oriented systems using dynamic analysis". In: *2011 International Workshop on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems*. Sept. 2011, pp. 1–5. DOI: 10.1109/MESOCA.2011.6049033 (cit. on pp. 45, 55, 93, 96).

[EZG14] T. Espinha, A. Zaidman, H. Gross. "Web API growing pains: Stories from client developers and their code". In: *2014 Software Evolution Week - IEEE Conference on Software Maintenance, Reengineering, and Reverse Engineering (CSMR-WCRE)*. Feb. 2014, pp. 84–93. DOI: 10.1109/CSMR-WCRE.2014.6747228 (cit. on pp. 49, 51, 52, 93–95).

[EZG15] T. Espinha, A. Zaidman, H.-G. Gross. "Web API growing pains: Loosely coupled yet strongly tied". In: *Journal of Systems and Software* 100 (2015), pp. 27–43. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2014.10.014. URL: http://www.sciencedirect.com/science/article/pii/S0164121214002180 (cit. on pp. 49, 51, 93–95).

[FBD+11] W. Fdhila, A. Baouab, K. Dahman, C. Godart, O. Perrin, F. Charoy. "Change propagation in decentralized composite web services". In: *7th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. Oct. 2011, pp. 508–511. DOI: 10.4108/icst.collaboratecom.2011.247097 (cit. on pp. 93, 96).

[FCH13] Z. Feng, D. K. W. Chiu, K. He. "A Service Evolution Registry with Alert-Based Management". In: *2013 Fifth International Conference on Service Science and Innovation*. May 2013, pp. 123–130. DOI: 10.1109/ICSSI.2013.33 (cit. on pp. 93, 96).

[Feu11] G. Feuerlicht. "Simple Metric for Assessing Quality of Service Design". In: *Service-Oriented Computing*. Ed. by E. M. Maximilien, G. Rossi, S.-T. Yuan, H. Ludwig, M. Fantinato. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 133–143. ISBN: 978-3-642-19394-1 (cit. on pp. 94, 95).

[FHH+13] Z. Feng, P. C. K. Hung, K. He, Y. Ma, M. Farwick, B. Li, R. Peng. "Towards a Taxonomy Framework of Evolution for SOA Solution: From a Practical Point of View". In: *Web Information Systems Engineering – WISE 2011 and 2012 Workshops*. Ed. by A. Haller, G. Huang, Z. Huang, H.-y. Paik, Q. Z. Sheng. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 261–274. ISBN: 978-3-642-38333-5 (cit. on pp. 94–96).

[FHPM11] Z. Feng, K. He, R. Peng, Y. Ma. "Taxonomy for Evolution of Service-Based System". In: *2011 IEEE World Congress on Services*. July 2011, pp. 331–338. DOI: 10.1109/SERVICES.2011.28 (cit. on pp. 94, 96).

[FL09] K. H. Fung, G. C. Low. "Methodology evaluation framework for dynamic evolution in composition-based distributed applications". In: *Journal of Systems and Software* 82.12 (2009), pp. 1950–1965. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2009.06.032. URL: http://www.sciencedirect.com/science/article/pii/S0164121209001393 (cit. on pp. 51, 94–96).

[FL13] J. L. Fiadeiro, A. Lopes. “A model for dynamic reconfiguration in service-oriented architectures”. In: *Software & Systems Modeling* 12.2 (May 2013), pp. 349–367. ISSN: 1619-1374. DOI: 10.1007/s10270-012-0236-1. URL: https://doi.org/10.1007/s10270-012-0236-1 (cit. on p. 31).

[FMT+11] M. Fokaefs, R. Mikhaiel, N. Tsantalis, E. Stroulia, A. Lau. “An Empirical Study on Web Service Evolution”. In: *2011 IEEE International Conference on Web Services*. July 2011, pp. 49–56. DOI: 10.1109/ICWS.2011.114 (cit. on pp. 93, 95).

[Fok14] M. Fokaefs. “WSDarwin: A Framework for the Support of Web Service Evolution”. In: *2014 IEEE International Conference on Software Maintenance and Evolution*. Sept. 2014, pp. 668–668. DOI: 10.1109/ICSME.2014.123 (cit. on pp. 51, 94, 96).

[Fok15] M.-E. Fokaefs. “WSDarwin: A Comprehensive Framework for Supporting Service-Oriented Systems Evolution”. In: (2015). DOI: 10.7939/r34746x4j. URL: https://era.library.ualberta.ca/items/2e927167-dae0-4b90-af67-b98ac37d0dfd (cit. on pp. 94–96).

[FOS15] M. Fokaefs, M. Oprescu, E. Stroulia. “WSDarwin: A web application for the support of REST service evolution”. In: *2015 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Sept. 2015, pp. 336–338. DOI: 10.1109/ICSM.2015.7332483 (cit. on pp. 94, 96).

[Fowa] M. Fowler. *Microservices*. https://martinfowler.com/articles/microservices.html?fbclid=IwAR34aOrLJz2VFZ92xyqXDTEw0Om7EiBl27MoQ5eRrNqCfnK7F8_LlwLK-bI. (Accessed on 03/26/2019). (Cit. on pp. 20, 21, 40).

[Fowb] M. Fowler. *PolyglotPersistence*. https://martinfowler.com/bliki/PolyglotPersistence.html. (Accessed on 03/26/2019). (Cit. on p. 20).

[Fow05] M. Fowler. *ServiceOrientedAmbiguity*. https://martinfowler.com/bliki/ServiceOrientedAmbiguity.html. (Accessed on 04/30/2019). July 2005 (cit. on p. 21).

[Fow08] M. Fowler. *ModelDrivenSoftwareDevelopment*. https://martinfowler.com/bliki/ModelDrivenSoftwareDevelopment.html. (Accessed on 05/17/2019). July 2008 (cit. on p. 37).

[FS12] M. Fokaefs, E. Stroulia. “WSDarwin: Automatic Web Service Client Adaptation”. In: *Proceedings of the 2012 Conference of the Center for Advanced Studies on Collaborative Research*. CASCON ’12. Toronto, Ontario, Canada: IBM Corp., 2012, pp. 176–191. URL: http://dl.acm.org/citation.cfm?id=2399776.2399792 (cit. on pp. 94–96).

[FS13a] M. Fokaefs, E. Stroulia. “WSDARWIN: A Decision-Support Tool for Web-Service Evolution”. In: *2013 IEEE International Conference on Software Maintenance*. Sept. 2013, pp. 444–447. DOI: 10.1109/ICSM.2013.68 (cit. on pp. 94, 96).

[FS13b] M. Fokaefs, E. Stroulia. “WSMeta: A Meta-model for Web Services to Compare Service Interfaces”. In: *Proceedings of the 17th Panhellenic Conference on Informatics*. PCI ’13. Thessaloniki, Greece: ACM, 2013, pp. 1–8. ISBN: 978-1-4503-1969-0. DOI: 10.1145/2491845.2491860. URL: http://doi.acm.org/10.1145/2491845.2491860 (cit. on pp. 94–96).

[FS14a] M. Fokaefs, E. Stroulia. “The WSDarwin Toolkit for Service-Client Evolution”. In: *2014 IEEE International Conference on Web Services*. June 2014, pp. 716–719. DOI: 10.1109/ICWS.2014.113 (cit. on pp. 94–96).

[FS14b] M. Fokaefs, E. Stroulia. “WSDarwin: Studying the Evolution of Web Service Systems”. In: *Advanced Web Services*. Ed. by A. Bouguettaya, Q. Z. Sheng, F. Daniel. New York, NY: Springer New York, 2014, pp. 199–223. ISBN: 978-1-4614-7535-4. DOI: 10.1007/978-1-4614-7535-4_9. URL: https://doi.org/10.1007/978-1-4614-7535-4_9 (cit. on pp. 94–96).

[FS16] M. Fokaefs, E. Stroulia. “Software Evolution in Web-Service Ecosystems: A Game-Theoretic Model”. In: *Service Science* 8.1 (2016), pp. 1–18. DOI: 10.1287/serv.2015.0114. eprint: https://doi.org/10.1287/serv.2015.0114. URL: https://doi.org/10.1287/serv.2015.0114 (cit. on pp. 94–96).

[FSM14] M. Fokaefs, E. Stroulia, P. R. Messinger. “Chapter 11 - Software Evolution in the Presence of Externalities: A Game-Theoretic Approach”. In: *Economics-Driven Software Architecture*. Ed. by I. Mistrik, R. Bahsoon, R. Kazman, Y. Zhang. Boston: Morgan Kaufmann, 2014, pp. 243–258. ISBN: 978-0-12-410464-8. DOI: https://doi.org/10.1016/B978-0-12-410464-8.00011-8. URL: http://www.sciencedirect.com/science/article/pii/B9780124104648000118 (cit. on pp. 94, 96).

[FTLL14] L. Fan, J. Tang, Y. Ling, B. Li. “Dynamic and Quantitative Method of Analyzing Service Consistency Evolution Based on Extended Hierarchical Finite State Automata”. In: *The Scientific World Journal* 2014 (2014), pp. 1–11. DOI: 10.1155/2014/793271. URL: https://doi.org/10.1155/2014/793271 (cit. on pp. 93, 95, 96).

[Fun11] K. H. Fung. “A method engineering approach to support dynamic evolution in composition-based distributed applications”. PhD thesis. Ph. D. Thesis, University of New South Wales, Sydney, Australia, 2011 (cit. on pp. 94, 96).

[GA11] M. Gebhart, S. Abeck. “Metrics for Evaluating Service Designs based on SoaML”. In: *International Journal on Advances in Software* 4 (Jan. 2011), pp. 61–75. URL: http://www.iariajournals.org/software/soft_v4_n12_2011_paged.pdf (cit. on pp. 94, 95).

[GBO+10] M. Gebhart, M. Baumgartner, S. Oehlert, M. Blersch, S. Abeck. “Evaluation of Service Designs Based on SoaML”. In: *2010 Fifth International Conference on Software Engineering Advances*. Aug. 2010, pp. 7–13. DOI: 10.1109/ICSEA.2010.8 (cit. on pp. 94–96).

[GCF+17] G. Granchelli, M. Cardarelli, P. D. Francesco, I. Malavolta, L. Iovino, A. D. Salle. “Towards Recovering the Software Architecture of Microservice-Based Systems”. In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)*. Apr. 2017, pp. 46–53. DOI: 10.1109/ICSAW.2017.48 (cit. on pp. 45, 55, 93, 95, 96).

[GJ14] P. K. Goyal, G. Joshi. “QMOOD metric sets to assess quality of Java program”. In: *2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)*. Feb. 2014, pp. 520–533. DOI: 10.1109/ICICICT.2014.6781337 (cit. on p. 50).

[GL09] Q. Gu, P. Lago. "Exploring service-oriented system engineering challenges: a systematic literature review". In: *Service Oriented Computing and Applications* 3.3 (Sept. 2009), pp. 171–188. ISSN: 1863-2394. DOI: 10.1007/s11761-009-0046-7. URL: https://doi.org/10.1007/s11761-009-0046-7 (cit. on p. 26).

[GL11] A. Goeb, K. Lochmann. "A Software Quality Model for SOA". In: *Proceedings of the 8th International Workshop on Software Quality*. WoSQ '11. Szeged, Hungary: ACM, 2011, pp. 18–25. ISBN: 978-1-4503-0851-9. DOI: 10.1145/2024587.2024593. URL: http://doi.acm.org/10.1145/2024587.2024593 (cit. on pp. 94, 96).

[GMK+11] A. Gehlert, A. Metzger, D. Karastoyanova, R. Kazhamiakin, K. Pohl, F. Leymann, M. Pistore. "Integrating Perfective and Corrective Adaptation of Service-based Applications". In: *Service Engineering: European Research Results*. Vienna: Springer Vienna, 2011, pp. 137–169. ISBN: 978-3-7091-0415-6. DOI: 10.1007/978-3-7091-0415-6_6. URL: https://doi.org/10.1007/978-3-7091-0415-6_6 (cit. on pp. 94–96).

[Hir12] M. A. Hirzalla. "Service Oriented Architecture Metrics Suite for Assessing and Predicting Business Agility Results of SOA Solutions". PhD thesis. DePaul University, June 2012. URL: http://sarec.nd.edu/Dissertations/MamounDissertation.pdf (cit. on pp. 94–96).

[HLK08] J. S. Her, H. J. La, S. D. Kim. "A Formal Approach to Devising a Practical Method for Modeling Reusable Services". In: *2008 IEEE International Conference on e-Business Engineering*. Oct. 2008, pp. 221–228. DOI: 10.1109/ICEBE.2008.20 (cit. on pp. 48, 93, 95, 96).

[HLM+11] W. Hummer, P. Leitner, A. Michlmayr, F. Rosenberg, S. Dustdar. "VRESCo – Vienna Runtime Environment for Service-oriented Computing". In: *Service Engineering: European Research Results*. Vienna: Springer Vienna, 2011, pp. 299–324. ISBN: 978-3-7091-0415-6. DOI: 10.1007/978-3-7091-0415-6_11. URL: https://doi.org/10.1007/978-3-7091-0415-6_11 (cit. on pp. 94–96).

[HO10] A. Hock-koon, M. Oussalah. "Defining Metrics for Loose Coupling Evaluation in Service Composition". In: *2010 IEEE International Conference on Services Computing*. July 2010, pp. 362–369. DOI: 10.1109/SCC.2010.17 (cit. on pp. 94, 95).

[HPD+14] R. S. Huergo, P. F. Pires, F. C. Delicato, B. Costa, E. Cavalcante, T. Batista. "A systematic survey of service identification methods". In: *Service Oriented Computing and Applications* 8.3 (Sept. 2014), pp. 199–219. ISSN: 1863-2394. DOI: 10.1007/s11761-014-0161-y. URL: https://doi.org/10.1007/s11761-014-0161-y (cit. on pp. 47, 93, 95).

[HW08] H. Hofmeister, G. Wirtz. "Supporting Service-Oriented Design with Metrics". In: *2008 12th International IEEE Enterprise Distributed Object Computing Conference*. Sept. 2008, pp. 191–200. DOI: 10.1109/EDOC.2008.13 (cit. on pp. 94, 95).

[HYX+11] J. Huang, H. Yang, L. Xu, B. Xu, H. Zhang. "Supporting context — Aware service evolution with a process management requirements model". In: *2011 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. Dec. 2011, pp. 1–8. DOI: 10.1109/SOCA.2011.6166229 (cit. on pp. 94–96).

[HZ12] B. Heinrich, S. Zimmermann. “Granularity Metrics for IT Services”. In: *33rd INTERNTIONAL CONFERENCE ON INFORMATION SYSTEMS (ICIS)*. Joey, F. George (Eds.), Proceedings of the International Conference on Information Systems, ICIS 2012, Orlando, USA, December 16-19, 2012. Association for Information Systems 2012, ISBN 978-0-615-71843-9. 2012, pp. 1–19. URL: https://epub.uni-regensburg.de/27167/ (cit. on pp. 94, 95).

[HZC11] M. A. Hirzalla, A. Zisman, J. Cleland-Huang. “Using Traceability to Support SOA Impact Analysis”. In: *2011 IEEE World Congress on Services*. July 2011, pp. 145–152. DOI: 10.1109/SERVICES.2011.103 (cit. on pp. 93, 95, 96).

[IAKK16] R. Ibrahim, K. Akil, A. Kalakech, S. Kadry. “A New Solution for Service Oriented Architecture Versioning”. In: *Asian Journal of Computer Science and Information Technology* 6 (July 2016) (cit. on pp. 94, 96).

[IHL+13] C. Inzinger, W. Hummer, I. Lytra, P. Leitner, H. Tran, U. Zdun, S. Dustdar. “Decisions, Models, and Monitoring – A Lifecycle Model for the Evolution of Service-Based Systems”. In: *2013 17th IEEE International Enterprise Distributed Object Computing Conference*. Sept. 2013, pp. 185–194. DOI: 10.1109/EDOC.2013.29 (cit. on pp. 39, 94–96).

[Ing18] J. Ingeno. *Software Architect’s Handbook*. Packt Publishing Ltd., 2018. ISBN: 9781788624060. URL: https://www.packtpub.com/application-development/software-architects-handbook (cit. on p. 24).

[ISO] ISO. *ISO 25010*. https://iso25000.com/index.php/en/iso-25000-standards/iso-25010?limit=3&start=6. (Accessed on 04/10/2019). (Cit. on pp. 22, 23).

[Jel15] J. Jelschen. “Service-oriented toolchains for software evolution”. In: *2015 IEEE 9th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)*. Oct. 2015, pp. 51–58. DOI: 10.1109/MESOCA.2015.7328127 (cit. on pp. 94–96).

[JGV+14] M. A. Jiménez, Á. V. Gómez, N. M. Villegas, G. Tamura, L. Duchien. “A Framework for Automated and Composable Testing of Component-Based Services”. In: *2014 IEEE 8th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems*. Sept. 2014, pp. 1–10. DOI: 10.1109/MESOCA.2014.9 (cit. on p. 31).

[JTKB18] M. I. Josélyne, D. Tuheirwe-Mukasa, B. Kanagwa, J. Balikuddembe. “Partitioning Microservices: A Domain Engineering Approach”. In: *Proceedings of the 2018 International Conference on Software Engineering in Africa*. SEiA ’18. Gothenburg, Sweden: ACM, 2018, pp. 43–49. ISBN: 978-1-4503-5719-7. DOI: 10.1145/3195528.3195535. URL: http://doi.acm.org/10.1145/3195528.3195535 (cit. on pp. 47, 48, 56, 93, 95, 96).

[K C13] and K. Chalmers and. “Evolution feature oriented Model Driven product line engineering approach for synergistic and dynamic service Evolution in Clouds: AO4BPEL3.0 proposal”. In: *International Conference on Information Society (i-Society 2013)*. June 2013, pp. 244–249 (cit. on pp. 45, 93, 96).

[KAR+11] A. Kazemi, A. N. Azizkandi, A. Rostampour, H. Haghighi, P. Jamshidi, F. Shams. "Measuring the Conceptual Coupling of Services Using Latent Semantic Indexing". In: *2011 IEEE International Conference on Services Computing*. July 2011, pp. 504–511. DOI: 10.1109/SCC.2011.23 (cit. on p. 94).

[KC07] B. Kitchenham, S. Charters. *Guidelines for performing Systematic Literature Reviews in Software Engineering*. 2007 (cit. on p. 27).

[KCH11] M. Kapuruge, A. Colman, J. Han. "Defining Customizable Business Processes without Compromising the Maintainability in Multi-tenant SaaS Applications". In: *2011 IEEE 4th International Conference on Cloud Computing*. July 2011, pp. 748–749. DOI: 10.1109/CLOUD.2011.93 (cit. on pp. 94, 96).

[KD08] N. Kulkarni, V. Dwivedi. "The Role of Service Granularity in a Successful SOA Realization A Case Study". In: *2008 IEEE Congress on Services - Part I*. July 2008, pp. 423–430. DOI: 10.1109/SERVICES-1.2008.86 (cit. on pp. 48, 93, 95, 96).

[KG12a] T. Karhikeyan, J. Geetha. "A metrics suite and fuzzy model for measuring coupling in Service Oriented Architecture". In: *2012 International Conference on Recent Advances in Computing and Software Systems*. Apr. 2012, pp. 254–259. DOI: 10.1109/RACSS.2012.6212677 (cit. on pp. 94, 96).

[KG12b] T. Karthikeyan, J. Geetha. "A Quantitative Measurement and Validation of Granularity in Service Oriented Architecture". In: vol. 9. 2. 2012, pp. 377–382. URL: http://www.ijcsi.org/articles/A-quantitative-measurement-and-validation-of-granularity-in-service-oriented-architecture.php (cit. on pp. 94, 95).

[KG12c] T. Karthikeyan, J. Geetha. "A Study and Critical Survey on Service Reusability Metrics". In: *International Journal of Information Technology and Computer Science* 4.5 (May 2012), pp. 25–31. DOI: 10.5815/ijitcs.2012.05.04. URL: https://doi.org/10.5815/ijitcs.2012.05.04 (cit. on pp. 94–96).

[KKCR09a] T. Kohlborn, A. Korthaus, T. Chan, M. Rosemann. "Identification and Analysis of Business and Software Services—A Consolidated Approach". In: *IEEE Transactions on Services Computing* 2.1 (Jan. 2009), pp. 50–64. ISSN: 1939-1374. DOI: 10.1109/TSC.2009.6 (cit. on pp. 93, 95, 96).

[KKCR09b] T. Kohlborn, A. Korthaus, T. Chan, M. Rosemann. "Service analysis : A critical assessment of the state of the art". In: *Proceedings of 17th European Conference on Information Systems. Information Systems in a Globalising World : Challenges, Ethics and Practices*. Ed. by S. Newell, E. Whitley, N. Pouloudi, J. Wareham, L. Mathiassen. Verona, Italy, June 2009. URL: https://eprints.qut.edu.au/26755/ (cit. on pp. 47, 93, 95).

[KKR17a] L. Kumar, A. Krishna, S. K. Rath. "The impact of feature selection on maintainability prediction of service-oriented applications". In: *Service Oriented Computing and Applications* 11.2 (June 2017), pp. 137–161. ISSN: 1863-2394. DOI: 10.1007/s11761-016-0202-9. URL: https://doi.org/10.1007/s11761-016-0202-9 (cit. on pp. 94, 95).

[KKR17b] L. Kumar, M. Kumar, S. K. Rath. "Maintainability prediction of web service using support vector machine with various kernel methods". In: *International Journal of System Assurance Engineering and Management* 8.2 (June 2017), pp. 205–222. ISSN: 0976-4348. DOI: 10.1007/s13198-016-0415-5. URL: https://doi.org/10.1007/s13198-016-0415-5 (cit. on pp. 94, 95).

[KLS07] M. Kajko-Mattsson, G. A. Lewis, D. B. Smith. “A Framework for Roles for Development, Evolution and Maintenance of SOA-Based Systems”. In: *Proceedings of the International Workshop on Systems Development in SOA Environments*. SDSOA ’07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 7–. ISBN: 0-7695-2960-7. DOI: 10.1109/SDSOA.2007.1. URL: http://dx.doi.org/10.1109/SDSOA.2007.1 (cit. on pp. 51, 94, 96).

[KLS08] M. Kajko-Mattsson, G. A. Lewis, D. B. Smith. “Evolution and Maintenance of SOA-Based Systems at SAS”. In: *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*. Jan. 2008, pp. 119–119. DOI: 10.1109/HICSS.2008.154 (cit. on pp. 39, 94–96).

[KMN+13] J. Kress, B. Maier, H. Normann, D. Schmeidel, G. Schmutz, B. Trops, C. Utschig-Utschig, T. Winterberg. *Enterprise Service Bus*. https://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html. (Accessed on 04/30/2019). July 2013 (cit. on p. 18).

[Koh08] T. Kohlborn. “A consolidated approach for service analysis”. PhD thesis. Westfalische-Wilhelms Universitat Munster, Mar. 2008. URL: https://eprints.qut.edu.au/13682/ (cit. on pp. 48, 93, 96).

[KR16] P. Kumar, Ratneshwer. “Some Observations on Dependency Analysis of SOA Based Systems”. In: *International Journal of Information Technology and Computer Science* 8.1 (Jan. 2016), pp. 54–66. DOI: 10.5815/ijitcs.2016.01.07. URL: https://doi.org/10.5815/ijitcs.2016.01.07 (cit. on pp. 94, 95).

[KRN+11] A. Kazemi, A. Rostampour, A. Nasirzadeh Azizkandi, H. Haghighi, F. Shams. “A metric suite for measuring service modularity”. In: *2011 CSI International Symposium on Computer Science and Software Engineering (CSSE)*. June 2011, pp. 95–102. DOI: 10.1109/CSICSSE.2011.5963997 (cit. on pp. 39, 94, 95).

[KRS17] L. Kumar, S. K. Rath, A. Sureka. “Using Source Code Metrics and Multivariate Adaptive Regression Splines to Predict Maintainability of Service Oriented Software”. In: *2017 IEEE 18th International Symposium on High Assurance Systems Engineering (HASE)*. Jan. 2017, pp. 88–95. DOI: 10.1109/HASE.2017.11 (cit. on pp. 50, 94, 95).

[KS18] L. Kumar, A. Sureka. “An Empirical Analysis on Web Service Anti-pattern Detection Using a Machine Learning Framework”. In: *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 01. July 2018, pp. 2–11. DOI: 10.1109/COMPSAC.2018.00010 (cit. on pp. 93, 95).

[KSB+11] H. Koziolek, B. Schlich, C. Bilich, R. Weiss, S. Becker, K. Krogmann, M. Trifu, R. Mirandola, A. Koziolek. “An Industrial Case Study on Quality Impact Prediction for Evolving Service-oriented Software”. In: *Proceedings of the 33rd International Conference on Software Engineering*. ICSE ’11. Waikiki, Honolulu, HI, USA: ACM, 2011, pp. 776–785. ISBN: 978-1-4503-0445-0. DOI: 10.1145/1985793.1985902. URL: http://doi.acm.org/10.1145/1985793.1985902 (cit. on pp. 94–96).

[KUSM18] M. Kleehaus, Ö. Uludağ, P. Schäfer, F. Matthes. “MICROLYZE: A Framework for Recovering the Software Architecture in Microservice-Based Environments”. In: *Information Systems in the Big Data Era*. Ed. by J. Mendling, H. Mouratidis. Cham: Springer International Publishing, 2018, pp. 148–162. ISBN: 978-3-319-92901-9 (cit. on pp. 37, 45, 55, 93, 95, 96).

[KZ13] S. Kijas, A. Zalewski. “Towards Evolution Methodology for Service-Oriented Systems”. In: *New Results in Dependability and Computer Systems*. Ed. by W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, J. Kacprzyk. Heidelberg: Springer International Publishing, 2013, pp. 255–273. ISBN: 978-3-319-00945-2 (cit. on pp. 93–96).

[Lar08] X. Larrucea. “MDSOA quality evaluation”. In: *2008 IEEE International Technology Management Conference (ICE)*. June 2008, pp. 1–6 (cit. on pp. 45, 93, 96).

[LCZ12] Y. Liu, W. Cazzola, B. Zhang. “Towards a Colored Reflective Petri-net Approach to Model Self-evolving Service-oriented Architectures”. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. SAC ’12. Trento, Italy: ACM, 2012, pp. 1858–1865. ISBN: 978-1-4503-0857-1. DOI: 10.1145/2245276.2232081. URL: http://doi.acm.org/10.1145/2245276.2232081 (cit. on pp. 37, 45, 46, 93, 95, 96).

[LEMP07] L. Lambers, H. Ehrig, L. Mariani, M. Pezzè. “Iterative Model-driven Development of Adaptable Service-based Applications”. In: *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*. ASE ’07. Atlanta, Georgia, USA: ACM, 2007, pp. 453–456. ISBN: 978-1-59593-882-4. DOI: 10.1145/1321631.1321707. URL: http://doi.acm.org/10.1145/1321631.1321707 (cit. on pp. 93, 96).

[LM12] H. Leopold, J. Mendling. “Automatic Derivation of Service Candidates from Business Process Model Repositories”. In: *Business Information Systems*. Ed. by W. Abramowicz, D. Kriksciuniene, V. Sakalauskas. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 84–95. ISBN: 978-3-642-30359-3 (cit. on pp. 48, 93, 95, 96).

[LMN10] J. Lee, D. Muthig, M. Naab. “A feature-oriented approach for developing reusable product line assets of service-based systems”. In: *Journal of Systems and Software* 83.7 (2010). SPLC 2008, pp. 1123–1136. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2010.01.048. URL: http://www.sciencedirect.com/science/article/pii/S0164121210000324 (cit. on pp. 39, 48, 93, 96).

[LMZ09] M. Liu, D. Ma, Y. Zhao. “An Approach to Identifying Conversation Dependency in Service Oriented System During Dynamic Evolution”. In: *Proceedings of the 2009 ACM Symposium on Applied Computing*. SAC ’09. Honolulu, Hawaii: ACM, 2009, pp. 1072–1073. ISBN: 978-1-60558-166-8. DOI: 10.1145/1529282.1529517. URL: http://doi.acm.org/10.1145/1529282.1529517 (cit. on pp. 93, 96).

[LMZS09] M. Liu, D. Ma, Y. Zhao, D. Sun. “An Approach to Preserving Consistency of SOAs in Dynamic Evolution”. In: *2009 Fourth International Conference on Internet and Web Applications and Services*. May 2009, pp. 505–509. DOI: 10.1109/ICIW.2009.81 (cit. on pp. 94, 96).

[LPM15] H. Leopold, F. Pittke, J. Mendling. “Automatic service derivation from business process model repositories via semantic technology”. In: *Journal of Systems and Software* 108 (2015), pp. 134–147. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2015.06.007. URL: http://www.sciencedirect.com/science/article/pii/S0164121215001235 (cit. on pp. 48, 93, 95, 96).

[LRRA11] M. Leotta, F. Ricca, G. Reggio, E. Astesiano. “Comparing the Maintainability of Two Alternative Architectures of a Postal System: SOA vs. Non-SOA”. In: *2011 15th European Conference on Software Maintenance and Reengineering*. Mar. 2011, pp. 317–320. DOI: 10.1109/CSMR.2011.41 (cit. on pp. 49, 93, 95).

[LS] G. A. Lewis, D. B. Smith. “Research Challenges in the Maintenance and Evolution of Service-Oriented Systems”. In: *Migrating Legacy Applications*. IGI Global, pp. 13–39. DOI: 10.4018/978-1-4666-2488-7.ch002. URL: https://doi.org/10.4018/978-1-4666-2488-7.ch002 (cit. on p. 94).

[LS07] G. Lewis, D. B. Smith. “Developing Realistic Approaches for the Migration of Legacy Components to Service-Oriented Architecture Environments”. In: *Trends in Enterprise Application Architecture*. Ed. by D. Draheim, G. Weber. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 226–240. ISBN: 978-3-540-75912-6 (cit. on p. 31).

[LS08] G. A. Lewis, D. B. Smith. “Service-Oriented Architecture and its implications for software maintenance and evolution”. In: *2008 Frontiers of Software Maintenance*. Sept. 2008, pp. 1–10. DOI: 10.1109/FOSM.2008.4659243 (cit. on pp. 52, 94).

[LWD08] S. J. Lowry, P. B. Warner, E. Deaubl. “NOAO Imaging Meta Data Quality Improvement: A Case Study of the Evolution of a Service Oriented System”. In: *Companion to the 23rd ACM SIGPLAN Conference on Object-oriented Programming Systems Languages and Applications*. OOPSLA Companion ’08. Nashville, TN, USA: ACM, 2008, pp. 675–684. ISBN: 978-1-60558-220-7. DOI: 10.1145/1449814.1449820. URL: http://doi.acm.org/10.1145/1449814.1449820 (cit. on pp. 38, 94–96).

[LXLZ13] J. Li, Y. Xiong, X. Liu, L. Zhang. “How Does Web Service API Evolution Affect Clients?” In: *2013 IEEE 20th International Conference on Web Services*. June 2013, pp. 300–307. DOI: 10.1109/ICWS.2013.48 (cit. on pp. 93, 95).

[MB10] Z. Ma, K. Ben. “Research on maintainability evaluation of service-oriented software”. In: *2010 3rd International Conference on Computer Science and Information Technology*. Vol. 4. July 2010, pp. 510–512. DOI: 10.1109/ICCSIT.2010.5563562 (cit. on pp. 94, 96).

[MJ11] H. Mu, S. Jiang. “Design patterns in software development”. In: *2011 IEEE 2nd International Conference on Software Engineering and Service Science*. July 2011, pp. 322–325. DOI: 10.1109/ICSESS.2011.5982228 (cit. on p. 37).

[MM14] A. A. Mohammed Elhag, R. Mohamad. “Metrics for evaluating the quality of service-oriented design”. In: *2014 8th. Malaysian Software Engineering Conference (MySEC)*. Sept. 2014, pp. 154–159. DOI: 10.1109/MySec.2014.6986006 (cit. on pp. 94, 95).

[MPN+12] N. Moha, F. Palma, M. Nayrolles, B. J. Conseil, Y.-G. Guéhéneuc, B. Baudry, J.-M. Jézéquel. "Specification and Detection of SOA Antipatterns". In: *Service-Oriented Computing*. Ed. by C. Liu, H. Ludwig, F. Toumani, Q. Yu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 1–16. ISBN: 978-3-642-34321-6 (cit. on pp. 93, 95, 96).

[MPS14] R. Mirandola, P. Potena, P. Scandurra. "Adaptation space exploration for service-oriented applications". In: *Science of Computer Programming* 80 (2014), pp. 356–384. ISSN: 0167-6423. DOI: https://doi.org/10.1016/j.scico.2013.09.017. URL: http://www.sciencedirect.com/science/article/pii/S0167642313002475 (cit. on pp. 46, 93, 94, 96).

[MSK10] T. Margaria, B. Steffen, C. Kubczak. "Evolution support in heterogeneous service-oriented landscapes". In: *Journal of the Brazilian Computer Society* 16.1 (May 2010), pp. 35–47. ISSN: 1678-4804. DOI: 10.1007/s13173-010-0004-4. URL: https://doi.org/10.1007/s13173-010-0004-4 (cit. on pp. 95, 96).

[Mul] Mulesoft. *Services in SOA | MuleSoft*. https://www.mulesoft.com/resources/esb/services-in-soa. (Accessed on 03/26/2019). (Cit. on p. 17).

[MW18] B. Mayer, R. Weinreich. "An Approach to Extract the Architecture of Microservice-Based Software Systems". In: *2018 IEEE Symposium on Service-Oriented System Engineering (SOSE)*. Mar. 2018, pp. 21–30. DOI: 10.1109/SOSE.2018.00012 (cit. on pp. 45, 55, 93, 95, 96).

[MYXK08] Z. Mingyan, W. Yanzhang, C. Xiaodong, X. Kai. "Service-Oriented Dynamic Evolution Model". In: *2008 International Symposium on Computational Intelligence and Design*. Vol. 1. Oct. 2008, pp. 322–326. DOI: 10.1109/ISCID.2008.147 (cit. on pp. 94–96).

[MZD11] C. Mayr, U. Zdun, S. Dustdar. "View-based model-driven architecture for enhancing maintainability of data access services". In: *Data Knowledge Engineering* 70.9 (2011), pp. 794–819. ISSN: 0169-023X. DOI: https://doi.org/10.1016/j.datak.2011.05.004. URL: http://www.sciencedirect.com/science/article/pii/S0169023X1100067X (cit. on pp. 93, 95, 96).

[NBM+15] M. Nayrolles, E. Beaudry, N. Moha, P. Valtchev, A. Hamou-Lhadj. "Towards Quality-Driven SOA Systems Refactoring Through Planning". In: *E-Technologies*. Ed. by M. Benyoucef, M. Weiss, H. Mili. Cham: Springer International Publishing, 2015, pp. 269–284. ISBN: 978-3-319-17957-5 (cit. on pp. 93, 95, 96).

[NGS15] K. Niklas, J. Greenyer, K. Schneider. "Towards Application and Evolution of Model-Based Heuristics for Improving SOA Service Design". In: *2015 IEEE/ACM 7th International Workshop on Modeling in Software Engineering*. May 2015, pp. 60–65. DOI: 10.1109/MiSE.2015.18 (cit. on pp. 38, 94–96).

[NM16] H. Naji, M. Mikki. "A Survey of Service Oriented Architecture Systems Maintenance Approaches". In: *International Journal of Computer Science and Information Technology* 8.3 (June 2016), pp. 21–29. DOI: 10.5121/ijcsit.2016.8302. URL: https://doi.org/10.5121/ijcsit.2016.8302 (cit. on pp. 94, 95).

[NMV13] M. Nayrolles, N. Moha, P. Valtchev. “Improving SOA antipatterns detection in Service Based Systems by mining execution traces”. In: *2013 20th Working Conference on Reverse Engineering (WCRE)*. Oct. 2013, pp. 321–330. DOI: 10.1109/WCRE.2013.6671307 (cit. on pp. 47, 93, 95, 96).

[NPMG13] M. Nayrolles, F. Palma, N. Moha, Y.-G. Guéhéneuc. “Soda: A Tool Support for the Detection of SOA Antipatterns”. In: *Service-Oriented Computing - ICSOC 2012 Workshops*. Ed. by A. Ghose, H. Zhu, Q. Yu, A. Delis, Q. Z. Sheng, O. Perrin, J. Wang, Y. Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 451–455. ISBN: 978-3-642-37804-1 (cit. on pp. 47, 93, 96).

[NW14] N. M. Nik Daud, W. M. N. Wan Kadir. “Static and dynamic classifications for SOA structural attributes metrics”. In: *2014 8th. Malaysian Software Engineering Conference (MySEC)*. Sept. 2014, pp. 130–135. DOI: 10.1109/MySec.2014.6986002 (cit. on pp. 94–96).

[ÖE12] P.-O. Östberg, E. Elmroth. “Service Development Abstraction: A Design Methodology and Development Toolset for Abstractive and Flexible Service-Based Software”. In: *Cloud Computing and Services Science*. Ed. by I. Ivanov, M. van Sinderen, B. Shishkov. New York, NY: Springer New York, 2012, pp. 165–184. ISBN: 978-1-4614-2326-3. DOI: 10.1007/978-1-4614-2326-3_9. URL: https://doi.org/10.1007/978-1-4614-2326-3_9 (cit. on pp. 94–96).

[OGKI15] A. Ouni, R. Gaikovina Kula, M. Kessentini, K. Inoue. “Web Service Antipatterns Detection Using Genetic Programming”. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO ’15. Madrid, Spain: ACM, 2015, pp. 1351–1358. ISBN: 978-1-4503-3472-3. DOI: 10.1145/2739480.2754724. URL: http://doi.acm.org/10.1145/2739480.2754724 (cit. on pp. 93, 95, 96).

[OKIC17] A. Ouni, M. Kessentini, K. Inoue, M. Ó. Cinnéide. “Search-Based Web Service Antipatterns Detection”. In: *IEEE Transactions on Services Computing* 10.4 (July 2017), pp. 603–617. ISSN: 1939-1374. DOI: 10.1109/TSC.2015.2502595 (cit. on pp. 93, 95, 96).

[OSIS16] A. Ouni, Z. Salem, K. Inoue, M. Soui. “SIM: An Automated Approach to Improve Web Service Interface Modularization”. In: *2016 IEEE International Conference on Web Services (ICWS)*. June 2016, pp. 91–98. DOI: 10.1109/ICWS.2016.20 (cit. on pp. 48, 93–96).

[PAK+14] F. Palma, L. An, F. Khomh, N. Moha, Y. Guéhéneuc. “Investigating the Change-Proneness of Service Patterns and Antipatterns”. In: *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*. Nov. 2014, pp. 1–8. DOI: 10.1109/SOCA.2014.43 (cit. on pp. 46, 93, 95).

[Pal13] F. Palma. “Detection of SOA Antipatterns”. In: *Service-Oriented Computing - ICSOC 2012 Workshops*. Ed. by A. Ghose, H. Zhu, Q. Yu, A. Delis, Q. Z. Sheng, O. Perrin, J. Wang, Y. Wang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 412–418. ISBN: 978-3-642-37804-1 (cit. on pp. 36, 37, 93, 95, 96).

[Pal15] F. Palma. “Unifying Service Oriented Technologies for the Specification and Detection of their Antipatterns”. PhD thesis. UNIVERSITÉ DE MONTRÉAL, 2015. URL: https://publications.polymtl.ca/1915/1/2015_FrancisPalma.pdf (cit. on pp. 47, 93, 95, 96).

[Pap03] M. P. Papazoglou. “Service-oriented computing: concepts, characteristics and directions”. In: *Proceedings of the Fourth International Conference on Web Information Systems Engineering, 2003. WISE 2003.* Dec. 2003, pp. 3–12. DOI: 10.1109/WISE.2003.1254461 (cit. on pp. 17, 18).

[PDMG14] F. Palma, J. Dubois, N. Moha, Y.-G. Guéhéneuc. “Detection of REST Patterns and Antipatterns: A Heuristics-Based Approach”. In: *Service-Oriented Computing*. Ed. by X. Franch, A. K. Ghose, G. A. Lewis, S. Bhiri. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 230–244. ISBN: 978-3-662-45391-9 (cit. on pp. 46, 47, 93, 95, 96).

[PGM+15] F. Palma, J. Gonzalez-Huerta, N. Moha, Y.-G. Guéhéneuc, G. Tremblay. “Are RESTful APIs Well-Designed? Detection of their Linguistic (Anti)Patterns”. In: *Service-Oriented Computing*. Ed. by A. Barros, D. Grigori, N. C. Narendra, H. K. Dam. Berlin, Heidelberg: Springer Berlin Heidelberg, 2015, pp. 171–187. ISBN: 978-3-662-48616-0 (cit. on pp. 47, 93, 95, 96).

[Pis17] J. K. V. Pismag. *Prediction of Web Service Antipatterns Using Machine Learning*. 2017. URL: http://hdl.handle.net/2027.42/136193 (cit. on pp. 47, 93, 95, 96).

[PK16] N. Parimala, R. Kohar. “A quality metric for BPEL process under evolution”. In: *2016 Eleventh International Conference on Digital Information Management (ICDIM)*. Sept. 2016, pp. 197–202. DOI: 10.1109/ICDIM.2016.7829777 (cit. on pp. 94, 95).

[PK18] N. Parimala, R. Kohar. “Evolution Metrics for a BPEL Process”. In: *Intelligent Computing and Information and Communication*. Ed. by S. Bhalla, V. Bhateja, A. A. Chandavale, A. S. Hiwale, S. C. Satapathy. Singapore: Springer Singapore, 2018, pp. 305–316. ISBN: 978-981-10-7245-1 (cit. on p. 94).

[PLW+10] C. Pichler, P. Langer, M. Wimmer, C. Huemer, B. Hofreiter. “Registry support for core component evolution”. In: *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. Dec. 2010, pp. 1–9. DOI: 10.1109/SOCA.2010.5707135 (cit. on pp. 94–96).

[PM15] F. Palma, N. Mohay. “A study on the taxonomy of service antipatterns”. In: *2015 IEEE 2nd International Workshop on Patterns Promotion and Anti-patterns Prevention (PPAP)*. Mar. 2015, pp. 5–8. DOI: 10.1109/PPAP.2015.7076848 (cit. on pp. 39, 47, 93, 96).

[PMH+15] A. Parvizi-Mosaed, S. Moaven, J. Habibi, G. Beigi, M. Naser-Shariat. “Towards a self-adaptive service-oriented methodology based on extended SOMA”. In: *Frontiers of Information Technology & Electronic Engineering* 16.1 (Jan. 2015), pp. 43–69. ISSN: 2095-9230. DOI: 10.1631/FITEE.1400040. URL: https://doi.org/10.1631/FITEE.1400040 (cit. on pp. 46, 93, 95, 96).

[PMTG14] F. Palma, N. Moha, G. Tremblay, Y.-G. Guéhéneuc. “Specification and Detection of SOA Antipatterns in Web Services”. In: *Software Architecture*. Ed. by P. Avgeriou, U. Zdun. Cham: Springer International Publishing, 2014, pp. 58–73. ISBN: 978-3-319-09970-5 (cit. on pp. 38, 55, 93, 95, 96).

[PNM+13] F. PALMA, M. NAYROLLES, N. MOHA, Y.-G. GUÉHÉNEUC, B. BAUDRY, J.-M. JÉZÉQUEL. "SOA ANTIPATTERNS: AN APPROACH FOR THEIR SPECIFICATION AND DETECTION". In: *International Journal of Cooperative Information Systems* 22.04 (2013), p. 1341004. DOI: 10.1142/S0218843013410049. eprint: https://doi.org/10.1142/S0218843013410049. URL: https://doi.org/10.1142/S0218843013410049 (cit. on pp. 93, 95, 96).

[PR11] M. Perepletchikov, C. Ryan. "A Controlled Experiment for Evaluating the Impact of Coupling on the Maintainability of Service-Oriented Software". In: *IEEE Transactions on Software Engineering* 37.4 (July 2011), pp. 449–465. ISSN: 0098-5589. DOI: 10.1109/TSE.2010.61 (cit. on pp. 30, 94, 95).

[PRF07] M. Perepletchikov, C. Ryan, K. Frampton. "Cohesion Metrics for Predicting Maintainability of Service-Oriented Software". In: *Seventh International Conference on Quality Software (QSIC 2007)*. Oct. 2007, pp. 328–335. DOI: 10.1109/QSIC.2007.4385516 (cit. on pp. 50, 94, 95).

[PRFT07] M. Perepletchikov, C. Ryan, K. Frampton, Z. Tari. "Coupling Metrics for Predicting Maintainability in Service-Oriented Designs". In: *2007 Australian Software Engineering Conference (ASWEC'07)*. Apr. 2007, pp. 329–340. DOI: 10.1109/ASWEC.2007.17 (cit. on pp. 50, 94, 95).

[PRT10] M. Perepletchikov, C. Ryan, Z. Tari. "The Impact of Service Cohesion on the Analyzability of Service-Oriented Software". In: *IEEE Transactions on Services Computing* 3.2 (Apr. 2010), pp. 89–103. ISSN: 1939-1374. DOI: 10.1109/TSC.2010.23 (cit. on pp. 94–96).

[QLJL14] D. Qiu, B. Li, S. Ji, H. Leung. "Regression Testing of Web Service: A Systematic Mapping Study". In: *ACM Comput. Surv.* 47.2 (Aug. 2014), 21:1–21:46. ISSN: 0360-0300. DOI: 10.1145/2631685. URL: http://doi.acm.org/10.1145/2631685 (cit. on p. 26).

[QLL+12] S. Qi, B. Li, C. Liu, X. Wu, R. Song. "A Trust Impact Analysis Model for Composite Service Evolution". In: *2012 19th Asia-Pacific Software Engineering Conference*. Vol. 1. Dec. 2012, pp. 73–78. DOI: 10.1109/APSEC.2012.30 (cit. on pp. 93, 96).

[Que] N. Quezada. *The 4 Types of Software Maintenance - Endertech*. https://endertech.com/blog/maintenance-bug-fixing-4-types-maintenance. (Accessed on 04/10/2019). (Cit. on p. 21).

[QX09] Z. Qingqing, L. Xinke. "Complexity Metrics for Service-Oriented Systems". In: *2009 Second International Symposium on Knowledge Acquisition and Modeling*. Vol. 3. Nov. 2009, pp. 375–378. DOI: 10.1109/KAM.2009.90 (cit. on pp. 50, 94).

[RCS+08] S. H. Ryu, F. Casati, H. Skogsrud, B. Benatallah, R. Saint-Paul. "Supporting the Dynamic Evolution of Web Service Protocols in Service-oriented Architectures". In: *ACM Trans. Web* 2.2 (May 2008), 13:1–13:46. ISSN: 1559-1131. DOI: 10.1145/1346337.1346241. URL: http://doi.acm.org/10.1145/1346337.1346241 (cit. on pp. 94–96).

[REW+11] T. Reichherzer, E. El-Sheikh, N. Wilde, L. White, J. Coffey, S. Simmons. “Towards intelligent search support for web services evolution identifying the right abstractions”. In: *2011 13th IEEE International Symposium on Web Systems Evolution (WSE)*. Sept. 2011, pp. 53–58. DOI: 10.1109/WSE.2011.6081819 (cit. on pp. 45, 93, 95, 96).

[RGR11] S. Rigby, V. Guana, F. Rueda. “A Highly Flexible and Light Mobile Service Oriented Architecture”. In: *Proceedings of the 9th International Conference on Advances in Mobile Computing and Multimedia*. MoMM ’11. Ho Chi Minh City, Vietnam: ACM, 2011, pp. 273–276. ISBN: 978-1-4503-0785-7. DOI: 10.1145/2095697.2095754. URL: http://doi.acm.org/10.1145/2095697.2095754 (cit. on pp. 39, 94, 96).

[Ric16] M. Richards. *Microservices vs. Service-Oriented Architecture*. O’Reilly Media, Inc, 2016. ISBN: 9781491975657 (cit. on p. 21).

[RKS+10] A. Rostampour, A. Kazemi, F. Shams, A. Zamiri, P. Jamshidi. “A metric for measuring the degree of entity-centric service cohesion”. In: *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. Dec. 2010, pp. 1–5. DOI: 10.1109/SOCA.2010.5707187 (cit. on pp. 94, 95).

[Rom15] D. Romano. “Analyzing the Change-Proneness of APIs and web APIs”. PhD thesis. Delft University of Technology, 2015. DOI: 10.4233/uuid:215065f7-d22d-4189-a5cb-0b2e91897cd1. URL: http://resolver.tudelft.nl/uuid:215065f7-d22d-4189-a5cb-0b2e91897cd1 (cit. on pp. 94–96).

[RP12] D. Romano, M. Pinzger. “Analyzing the Evolution of Web Services Using Fine-Grained Changes”. In: *2012 IEEE 19th International Conference on Web Services*. June 2012, pp. 392–399. DOI: 10.1109/ICWS.2012.29 (cit. on pp. 49, 93–96).

[RRS+11] J. Ramanathan, R. Ramnath, N. Singh, Z. Xu, Y. Xu. “Sense-respond Cloud Mediator Architecture for Services Evolution”. In: *Proceedings of the 2011 ACM Symposium on Applied Computing*. SAC ’11. TaiChung, Taiwan: ACM, 2011, pp. 162–169. ISBN: 978-1-4503-0113-8. DOI: 10.1145/1982185.1982223. URL: http://doi.acm.org/10.1145/1982185.1982223 (cit. on pp. 94–96).

[SAD16] W. Scarborough, C. Arnold, M. Dahan. “Case Study: Microservice Evolution and Software Lifecycle of the XSEDE User Portal API”. In: *Proceedings of the XSEDE16 Conference on Diversity, Big Data, and Science at Scale*. XSEDE16. Miami, USA: ACM, 2016, 47:1–47:5. ISBN: 978-1-4503-4755-6. DOI: 10.1145/2949550.2949655. URL: http://doi.acm.org/10.1145/2949550.2949655 (cit. on pp. 94, 95).

[SAM15] S. M. Sohan, C. Anslow, F. Maurer. “A Case Study of Web API Evolution”. In: *2015 IEEE World Congress on Services*. June 2015, pp. 245–252. DOI: 10.1109/SERVICES.2015.43 (cit. on pp. 49, 93, 95).

[SBB11] S. Slimani, S. Baïna, K. Baïna. “Interactive Ontology Evolution Management Using Mutli-agent System: A Proposal for Sustainability of Semantic Interoperability in SOA”. In: *2011 IEEE 20th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*. June 2011, pp. 41–46. DOI: 10.1109/WETICE.2011.15 (cit. on pp. 94–96).

[SCKP08] B. Shim, S. Choue, S. Kim, S. Park. “A Design Quality Model for Service-Oriented Architecture”. In: *2008 15th Asia-Pacific Software Engineering Conference*. Dec. 2008, pp. 403–410. DOI: 10.1109/APSEC.2008.32 (cit. on pp. 94–96).

[SFCK14] H. Sbai, M. Fredj, B. Chakir, L. Kjiri. "On managing evolution of configurable services based on configurable processes". In: *2014 Second World Conference on Complex Systems (WCCS)*. Nov. 2014, pp. 616–621. DOI: 10.1109/ICoCS.2014.7060934 (cit. on pp. 94–96).

[SHM+13] I. Solichah, M. Hamilton, P. Mursanto, C. Ryan, M. Perepletchikov. "Exploration on software complexity metrics for business process model and notation". In: *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*. Sept. 2013, pp. 31–37. DOI: 10.1109/ICACSIS.2013.6761549 (cit. on pp. 94, 95).

[SJ18] W. Song, H. Jacobsen. "Static and Dynamic Process Change". In: *IEEE Transactions on Services Computing* 11.1 (Jan. 2018), pp. 215–231. ISSN: 1939-1374. DOI: 10.1109/TSC.2016.2536025 (cit. on pp. 49, 93, 95).

[SKH+17] A. R. Sampaio, H. Kadiyala, B. Hu, J. Steinbacher, T. Erwin, N. Rosa, I. Beschastnikh, J. Rubin. "Supporting Microservice Evolution". In: *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. Sept. 2017, pp. 539–543. DOI: 10.1109/ICSME.2017.63 (cit. on pp. 45, 93, 95, 96).

[SLM12] V. E. S. Souza, A. Lapouchnian, J. Mylopoulos. "(Requirement) Evolution Requirements for Adaptive Systems". In: *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. SEAMS '12. Zurich, Switzerland: IEEE Press, 2012, pp. 155–164. ISBN: 978-1-4673-1787-0. URL: http://dl.acm.org/citation.cfm?id=2666795.2666820 (cit. on p. 51).

[SMSL17] I. Sofat, A. Mathrani, C. Scogings, B. Li. "Modular web service design for agility and maintainability with unity container". In: *2017 International Conference on Data Management, Analytics and Innovation (ICDMAI)*. Feb. 2017, pp. 259–263. DOI: 10.1109/ICDMAI.2017.8073521 (cit. on pp. 94–96).

[SP15] T. Senivongse, A. Puapolthep. "A maintainability assessment model for service-oriented systems". In: *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 1. 2015, pp. 139–144 (cit. on pp. 50, 94–96).

[SPR+19] F. Sabir, F. Palma, G. Rasool, Y.-G. Guéhéneuc, N. Moha. "A systematic literature review on the detection of smells and their evolution in object-oriented and service-oriented systems". In: *Software: Practice and Experience* 49.1 (2019), pp. 3–39. DOI: 10.1002/spe.2639. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/spe.2639. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2639 (cit. on pp. 47, 93, 95).

[SRY17] F. Sabir, G. Rasool, M. Yousaf. "A Lightweight Approach for Specification and Detection of SOAP Anti-Patterns". In: *International Journal of Advanced Computer Science and Applications* 8 (Jan. 2017). DOI: 10.14569/IJACSA.2017.080555 (cit. on pp. 93, 95, 96).

[SS09] R. Sindhgatta, B. Sengupta. "An Extensible Framework for Tracing Model Evolution in SOA Solution Design". In: *Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications*. OOPSLA '09. Orlando, Florida, USA: ACM, 2009, pp. 647–658. ISBN: 978-1-60558-768-4. DOI: 10.1145/1639950.1639960. URL: http://doi.acm.org/10.1145/1639950.1639960 (cit. on pp. 45, 93, 96).

[SSP09] R. Sindhgatta, B. Sengupta, K. Ponnalagu. “Measuring the Quality of Service Oriented Design”. In: *Service-Oriented Computing*. Ed. by L. Baresi, C.-H. Chi, J. Suzuki. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 485–499. ISBN: 978-3-642-10383-4 (cit. on pp. 30, 94, 95).

[Sta18] T. O. C. Staff. *SOA versus microservices: What's the difference? - Cloud computing news*. https://www.ibm.com/blogs/cloud-computing/2018/09/06/soa-versus-microservices/. (Accessed on 05/07/2019). Sept. 2018 (cit. on p. 22).

[STH18] J. Soldani, D. A. Tamburri, W.-J. V. D. Heuvel. “The pains and gains of microservices: A Systematic grey literature review”. In: *Journal of Systems and Software* 146 (2018), pp. 215–232. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2018.09.082. URL: http://www.sciencedirect.com/science/article/pii/S0164121218302139 (cit. on pp. 25, 61).

[SVP12] G. Shanmugasundaram, V. P. Venkatesan, C. Punitha Devi. “Research opportunities in service reusability of service oriented architecture”. In: *2012 International Conference on Emerging Trends in Science, Engineering and Technology (INCOSET)*. Dec. 2012, pp. 396–403. DOI: 10.1109/INCOSET.2012.6513940 (cit. on pp. 94–96).

[SZZ+12] W. Song, G. Zhang, Y. Zou, Q. Yang, X. Ma. “Towards Dynamic Evolution of Service Choreographies”. In: *2012 IEEE Asia-Pacific Services Computing Conference*. Dec. 2012, pp. 225–232. DOI: 10.1109/APSCC.2012.40 (cit. on pp. 94–96).

[TBK+16] H. T. Tran, H. Baraki, R. Kuppili, A. Taherkordi, K. Geihs. “A Notification Management Architecture for Service Co-evolution in the Internet of Things”. In: *2016 IEEE 10th International Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Environments (MESOCA)*. Oct. 2016, pp. 9–15. DOI: 10.1109/MESOCA.2016.8 (cit. on pp. 93, 96).

[TDKL18] B. Terzic, V. Dimitrieski, S. Kordić, I. Luković. “A Model-Driven Approach to Microservice Software Architecture Establishment”. In: *Position Papers of the 2018 Federated Conference on Computer Science and Information Systems*. PTI, Sept. 2018. DOI: 10.15439/2018f370. URL: https://doi.org/10.15439/2018f370 (cit. on pp. 62, 63).

[TJ14] M. Thirumaran, M. Jannani. “Theoretical foundation to evaluate the change measures for an effective web service change management”. In: *Proceedings of IEEE International Conference on Computer Communication and Systems ICCCS14*. Feb. 2014, pp. 226–232. DOI: 10.1109/ICCCS.2014.7068197 (cit. on pp. 93, 96).

[TJSD10] M. Treiber, L. Juszczyk, D. Schall, S. Dustdar. “Programming Evolvable Web Services”. In: *Proceedings of the 2Nd International Workshop on Principles of Engineering Service-Oriented Systems*. PESOS '10. Cape Town, South Africa: ACM, 2010, pp. 43–49. ISBN: 978-1-60558-963-3. DOI: 10.1145/1808885.1808895. URL: http://doi.acm.org/10.1145/1808885.1808895 (cit. on pp. 94–96).

[TL18] D. Taibi, V. Lenarduzzi. “On the Definition of Microservice Bad Smells”. In: *IEEE Software* 35.3 (May 2018), pp. 56–62. ISSN: 0740-7459. DOI: 10.1109/MS.2018.2141031 (cit. on pp. 47, 55, 93, 95).

[TNSC17] L. P. Tizzei, M. Nery, V. C. V. B. Segura, R. F. G. Cerqueira. "Using Microservices and Software Product Line Engineering to Support Reuse of Evolving Multi-tenant SaaS". In: *Proceedings of the 21st International Systems and Software Product Line Conference - Volume A*. SPLC '17. Sevilla, Spain: ACM, 2017, pp. 205–214. ISBN: 978-1-4503-5221-5. DOI: 10.1145/3106195.3106224. URL: http://doi.acm.org/10.1145/3106195.3106224 (cit. on pp. 94, 95).

[TSZ18] S. Tragatschnig, S. Stevanetic, U. Zdun. "Supporting the evolution of event-driven service-oriented architectures using change patterns". In: *Information and Software Technology* 100 (2018), pp. 133–146. ISSN: 0950-5849. DOI: https://doi.org/10.1016/j.infsof.2018.04.005. URL: http://www.sciencedirect.com/science/article/pii/S0950584916303251 (cit. on pp. 46, 93, 95, 96).

[TY13] Y. Tamura, S. Yamada. "Service-Oriented Maintainability Modeling and Analysis for a Cloud Computing". In: *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops*. July 2013, pp. 53–58. DOI: 10.1109/COMPSACW.2013.23 (cit. on pp. 94–96).

[VCB+18a] C. C. Venters, R. Capilla, S. Betz, B. Penzenstadler, T. Crick, S. Crouch, E. Y. Nakagawa, C. Becker, C. Carrillo. "Software sustainability: Research and practice from a software architecture viewpoint". In: *Journal of Systems and Software* 138 (2018), pp. 174–188. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2017.12.026. URL: http://www.sciencedirect.com/science/article/pii/S0164121217303072 (cit. on p. 25).

[VCB+18b] C. C. Venters, R. Capilla, S. Betz, B. Penzenstadler, T. Crick, S. Crouch, E. Y. Nakagawa, C. Becker, C. Carrillo. "Software sustainability: Research and practice from a software architecture viewpoint". In: *Journal of Systems and Software* 138 (2018), pp. 174–188. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2017.12.026. URL: http://www.sciencedirect.com/science/article/pii/S0164121217303072 (cit. on p. 30).

[VKG17] H. Vural, M. Koyuncu, S. Guney. "A Systematic Literature Review on Microservices". In: *Computational Science and Its Applications – ICCSA 2017*. Ed. by O. Gervasi, B. Murgante, S. Misra, G. Borruso, C. M. Torre, A. M. A. Rocha, D. Taniar, B. O. Apduhan, E. Stankova, A. Cuzzocrea. Cham: Springer International Publishing, 2017, pp. 203–217. ISBN: 978-3-319-62407-5 (cit. on p. 26).

[w3sa] w3schools. *RESTful Web Services*. https://www.w3schools.in/restful-web-services/intro/. (Accessed on 04/30/2019). (Cit. on p. 18).

[w3sb] w3schools. *XML Soap*. https://www.w3schools.com/xml/xml_soap.asp. (Accessed on 04/30/2019). (Cit. on p. 18).

[w3sc] w3schools. *XML Tutorial*. https://www.w3schools.com/xml/default.asp. (Accessed on 04/30/2019). (Cit. on p. 18).

[w3sd] w3schools. *XML WSDL*. https://www.w3schools.com/xml/xml_wsdl.asp. (Accessed on 04/30/2019). (Cit. on p. 17).

[Wag13] S. Wagner. *Software Product Quality Control*. Springer Berlin Heidelberg, 2013. DOI: 10.1007/978-3-642-38571-1. URL: https://doi.org/10.1007/978-3-642-38571-1 (cit. on pp. 15, 21).

[Wan09] X. Wang. “Metrics for Evaluating Coupling and Service Granularity in Service Oriented Architecture”. In: *2009 International Conference on Information Engineering and Computer Science*. Dec. 2009, pp. 1–4. DOI: 10.1109/ICIECS.2009.5362767 (cit. on pp. 94, 95).

[Wan18] H. Wang. “Intelligent Web Services Architecture Evolution Via An Automated Learning-Based Refactoring Framework”. PhD thesis. University of Michigan, 2018. URL: http://hdl.handle.net/2027.42/142810 (cit. on pp. 93–96).

[WB14a] R. Weinreich, G. Buchgeher. “Automatic Reference Architecture Conformance Checking for SOA-Based Software Systems”. In: *2014 IEEE/IFIP Conference on Software Architecture*. Apr. 2014, pp. 95–104. DOI: 10.1109/WICSA.2014.22 (cit. on p. 39).

[WB14b] R. Weinreich, G. Buchgeher. “Automatic Reference Architecture Conformance Checking for SOA-Based Software Systems”. In: *2014 IEEE/IFIP Conference on Software Architecture*. Apr. 2014, pp. 95–104. DOI: 10.1109/WICSA.2014.22 (cit. on pp. 45, 93–96).

[WC11] S. Wang, M. A. M. Capretz. “Dependency and Entropy Based Impact Analysis for Service-Oriented System Evolution”. In: *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*. Vol. 1. Aug. 2011, pp. 412–417. DOI: 10.1109/WI-IAT.2011.196 (cit. on pp. 93, 96).

[WC15] Z. Wang, G. Cheng. “An approach to synergistic and dynamic service evolution in clouds”. In: *International Journal of Cloud Computing* 4.2 (2015), p. 177. DOI: 10.1504/ijcc.2015.069275. URL: https://doi.org/10.1504/ijcc.2015.069275 (cit. on pp. 39, 94, 96).

[WCX10] Z. Wang, D. Chu, X. Xu. “Value Network Based Service Choreography Design and Evolution”. In: *2010 IEEE 7th International Conference on E-Business Engineering*. Nov. 2010, pp. 495–500. DOI: 10.1109/ICEBE.2010.13 (cit. on pp. 94–96).

[WGEZ16] N. Wilde, B. Gonen, E. El-Sheikh, A. Zimmermann. “Approaches to the Evolution of SOA Systems”. In: *Emerging Trends in the Evolution of Service-Oriented and Enterprise Architectures*. Ed. by E. El-Sheikh, A. Zimmermann, L. C. Jain. Cham: Springer International Publishing, 2016, pp. 5–21. ISBN: 978-3-319-40564-3. DOI: 10.1007/978-3-319-40564-3_2. URL: https://doi.org/10.1007/978-3-319-40564-3_2 (cit. on pp. 94, 96).

[WHHC14] S. Wang, W. A. Higashino, M. Hayes, M. A. M. Capretz. “Service Evolution Patterns”. In: *2014 IEEE International Conference on Web Services*. June 2014, pp. 201–208. DOI: 10.1109/ICWS.2014.39 (cit. on pp. 46, 55, 93).

[WJ12] Y.-C. Wu, H. C. Jiau. “A Monitoring Mechanism to Support Agility in Service-based Application Evolution”. In: *SIGSOFT Softw. Eng. Notes* 37.5 (Sept. 2012), pp. 1–10. ISSN: 0163-5948. DOI: 10.1145/2347696.2347714. URL: http://doi.acm.org/10.1145/2347696.2347714 (cit. on pp. 94, 96).

[WKO16] H. Wang, M. Kessentini, A. Ouni. “Prediction of Web Services Evolution”. In: *Service-Oriented Computing*. Ed. by Q. Z. Sheng, E. Stroulia, S. Tata, S. Bhiri. Cham: Springer International Publishing, 2016, pp. 282–297. ISBN: 978-3-319-46295-0 (cit. on pp. 49–51, 93–96).

[WKZ14] S. Wang, I. Keivanloo, Y. Zou. "How Do Developers React to RESTful API Evolution?" In: *Service-Oriented Computing*. Ed. by X. Franch, A. K. Ghose, G. A. Lewis, S. Bhiri. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 245–259. ISBN: 978-3-662-45391-9 (cit. on pp. 39, 49, 93, 95).

[WRC+13] L. J. White, T. Reichherzer, J. Coffey, N. Wilde, S. Simmons. "Maintenance of service oriented architecture composite applications: static and dynamic support". In: *Journal of Software: Evolution and Process* 25.1 (2013), pp. 97–109. DOI: 10.1002/smr.568. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/smr.568. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/smr.568 (cit. on pp. 94–96).

[WW13] Y. Wang, Y. Wang. "A survey of change management in service-based environments". In: *Service Oriented Computing and Applications* 7.4 (Dec. 2013), pp. 259–273. ISSN: 1863-2394. DOI: 10.1007/s11761-013-0128-4. URL: https://doi.org/10.1007/s11761-013-0128-4 (cit. on pp. 49, 51, 93–95).

[WW15] Y. Wang, Y. Wang. "Change Taxonomy and Service Importance Factor for Change Analysis in Composite Service". In: *2015 IEEE 12th International Conference on e-Business Engineering*. Oct. 2015, pp. 127–134. DOI: 10.1109/ICEBE.2015.30 (cit. on pp. 94, 96).

[WWR+12] L. White, N. Wilde, T. Reichherzer, E. El-Sheikh, G. Goehring, A. Baskin, B. Hartmann, M. Manea. "Understanding Interoperable Systems: Challenges for the Maintenance of SOA Applications". In: *2012 45th Hawaii International Conference on System Sciences*. Jan. 2012, pp. 2199–2206. DOI: 10.1109/HICSS.2012.614 (cit. on pp. 94–96).

[WYZ10] Y. Wang, J. Yang, W. Zhao. "Change impact analysis for service based business processes". In: *2010 IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*. Dec. 2010, pp. 1–8. DOI: 10.1109/SOCA.2010.5707155 (cit. on pp. 49, 93, 96).

[WYZ11] Y. Wang, J. Yang, W. Zhao. "A Change Analysis Tool for Service-Based Business Processes". In: *Web Information System Engineering – WISE 2011*. Ed. by A. Bouguettaya, M. Hauswirth, L. Liu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 336–337. ISBN: 978-3-642-24434-6 (cit. on pp. 39, 93, 96).

[WYZS12] Y. Wang, J. Yang, W. Zhao, J. Su. "Change impact analysis in service-based business processes". In: *Service Oriented Computing and Applications* 6.2 (June 2012), pp. 131–149. ISSN: 1863-2394. DOI: 10.1007/s11761-011-0093-8. URL: https://doi.org/10.1007/s11761-011-0093-8 (cit. on pp. 93, 96).

[Xa12] and X. Liu, K. C. and. "Evolution pattern for Service Evolution in Clouds". In: *2012 International Conference for Internet Technology and Secured Transactions*. Dec. 2012, pp. 704–709 (cit. on pp. 46, 93, 96).

[XGZ07] H. Xiao, J. Guo, Y. Zou. "Supporting Change Impact Analysis for Service Oriented Business Applications". In: *Proceedings of the International Workshop on Systems Development in SOA Environments*. SDSOA '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 6–. ISBN: 0-7695-2960-7. DOI: 10.1109/SDSOA.2007.11. URL: http://dx.doi.org/10.1109/SDSOA.2007.11 (cit. on pp. 93, 95, 96).

[XYCL17] H. Xie, J. Yang, C. K. Chang, L. Liu. “A statistical analysis approach to predict user’s changing requirements for software service evolution”. In: *Journal of Systems and Software* 132 (2017), pp. 147–164. ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2017.06.071. URL: http://www.sciencedirect.com/science/article/pii/S0164121217301358 (cit. on pp. 51, 94–96).

[XYZ17] P. Xiu, J. Yang, W. Zhao. “A Change Management Framework for Service Based Business Process”. In: *Proceedings of the Australasian Computer Science Week Multiconference*. ACSW ’17. Geelong, Australia: ACM, 2017, 36:1–36:9. ISBN: 978-1-4503-4768-6. DOI: 10.1145/3014812.3014850. URL: http://doi.acm.org/10.1145/3014812.3014850 (cit. on pp. 93, 96).

[XYZ19] P. Xiu, J. Yang, W. Zhao. “Change management of service-based business processes”. In: *Service Oriented Computing and Applications* 13.1 (Mar. 2019), pp. 51–66. ISSN: 1863-2394. DOI: 10.1007/s11761-018-0250-4. URL: https://doi.org/10.1007/s11761-018-0250-4 (cit. on pp. 93, 95, 96).

[You16] R. Yousef. “Assessing the Quality of Candidate Software Services Generated Using Service Identification Approaches”. In: *International Journal of Computer Science Issues (IJCSI)* 13.2 (2016), p. 64. DOI: 10.20943/01201602.6471. URL: https://doi.org/10.20943/01201602.6471 (cit. on pp. 48, 93, 95, 96).

[YVBG12] M. Yamashita, B. Vollino, K. Becker, R. Galante. “Measuring Change Impact Based on Usage Profiles”. In: *2012 IEEE 19th International Conference on Web Services*. June 2012, pp. 226–233. DOI: 10.1109/ICWS.2012.35 (cit. on pp. 93, 95, 96).

[ZAB15] W. Zuo, Y. Amghar, A. Benharkat. “The Impact Analysis Model for Web Service Evolution”. In: *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*. Vol. 1. Dec. 2015, pp. 457–460. DOI: 10.1109/WI-IAT.2015.199 (cit. on pp. 39, 93, 96).

[ZBA14a] W. Zuo, A. N. Benharkat, Y. Amghar. “Change-centric Model for Web Service Evolution”. In: *2014 IEEE International Conference on Web Services*. June 2014, pp. 712–713. DOI: 10.1109/ICWS.2014.111 (cit. on pp. 94–96).

[ZBA14b] W. Zuo, A. N. Benharkat, Y. Amghar. “Holistic and Change-centric Model for Web Service Evolution”. In: *2014 IEEE World Congress on Services*. June 2014, pp. 250–253. DOI: 10.1109/SERVICES.2014.51 (cit. on pp. 94, 96).

[ZDZ10] L. Zhang, V. Dwivedi, N. Zhou. “Formalizing ”Traceability’ for Architectural Evolutions”. In: *2010 IEEE Asia-Pacific Services Computing Conference*. Dec. 2010, pp. 285–292. DOI: 10.1109/APSCC.2010.118 (cit. on pp. 46, 93, 96).

[ZGZW16] Y. Zhou, J. Ge, P. Zhang, W. Wu. “Model based verification of dynamically evolvable service oriented systems”. In: *Science China Information Sciences* 59.3 (Jan. 2016), p. 32101. ISSN: 1869-1919. DOI: 10.1007/s11432-015-5332-8. URL: https://doi.org/10.1007/s11432-015-5332-8 (cit. on pp. 46, 93–96).

[ZHZ+16] L. Zhong, J. He, N. Zhang, P. Zhang, J. Xia. “Software Evolution Information Driven Service-Oriented Software Clustering”. In: *2016 IEEE International Congress on Big Data (BigData Congress)*. June 2016, pp. 493–500. DOI: 10.1109/BigDataCongress.2016.75 (cit. on pp. 94–96).

[ZJZ+16] X. Zhou, Y. Jin, H. Zhang, S. Li, X. Huang. "A Map of Threats to Validity of Systematic Literature Reviews in Software Engineering". In: *2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*. Dec. 2016, pp. 153–160. DOI: 10.1109/APSEC.2016.031 (cit. on p. 57).

[ZNL17] U. Zdun, E. Navarro, F. Leymann. "Ensuring and Assessing Architecture Conformance to Microservice Decomposition Patterns". In: *Service-Oriented Computing*. Ed. by M. Maximilien, A. Vallecillo, J. Wang, M. Oriol. Cham: Springer International Publishing, 2017, pp. 411–429. ISBN: 978-3-319-69035-3 (cit. on pp. 36, 46–48, 55, 56, 93, 95, 96).

[ZSK12] A. Zalewski, M. Szlenk, S. Kijas. "AN EVOLUTION PROCESS FOR SERVICE-ORIENTED SYSTEMS". In: *Computer Science* 13.4 (2012), p. 71. ISSN: 2300-7036. URL: https://journals.agh.edu.pl/csci/article/view/45 (cit. on pp. 94, 96).

[Zuo16] W. Zuo. "Managing and modeling web service evolution in SOA architecture". Theses. Université de Lyon, July 2016. URL: https://tel.archives-ouvertes.fr/tel-01694135 (cit. on pp. 94–96).

[ZYA14] W. Zuo, A. Youssef, B. Aïcha-Nabila. "Programming Framework based on change-centric web service evolution model". In: *The 4th International Symposium on Web Services (WSS'2014)*. Sfax, Tunisia, Dec. 2014. URL: https://hal.archives-ouvertes.fr/hal-01535808 (cit. on pp. 52, 94–96).

[ZYL11] S. Zhang, J. Yin, R. Liu. "A RGPS-based framework for service-oriented requirement evolution of networked software". In: *2011 IEEE 3rd International Conference on Communication Software and Networks*. May 2011, pp. 321–325. DOI: 10.1109/ICCSN.2011.6013724 (cit. on pp. 51, 94, 96).

All links were last followed on June 2, 2019.

A Full Thematical Categorization

Architecture Recovery and Documentation	**Model-driven Approaches**	**Patterns**
[BWH18], [MW18], [WB14b], **[KUSM18]**, **[SKH+17]**, [REW+11], **[AAE18]**, **[GCF+17]**, [EZG11]	[BABM11], [SS09], [EMK12], [K C13], [ZDZ10], [LEMP07], [Lar08], [ZGZW16], [TSZ18], [LCZ12], [DCG11], [MZD11]	[MPS14], [PDMG14], [DMT13], **[ZNL17]**, [Xa12], [ZDZ10], [PAK+14], [AP10], [AJP12], [Ath17], [WHHC14], [TSZ18], [PMH+15], [Bog18]

Table A.1: Thematical categorization 1/3, **microservices**, SOA, microservices and SOA

Antipatterns and Bad Smells	**Service Identification and Decomposition**	**Change Impact and Scenarios**
[SRY17], [PM15], [SPR+19], [KS18], [PGM+15], [PDMG14], [Pal13], [NMV13], [Wan18], [PAK+14], **[CBD18]**, **[TL18]**, [Pis17], [OKIC17], [PNM+13], [NPMG13], [MPN+12], [PMTG14], [NBM+15], [Pal15], [OGKI15]	[Koh08], [LMN10], [HLK08], [BB18a], [BB18b], [CLW11], [HPD+14], [AS16], [You16], [LM12], [LPM15], [AZM+15], **[ZNL17]**, [KKCR09a], **[JTKB18]**, [KKCR09b], [EKM07], [BCDP14], [OSIS16], [KD08], [DOK+17]	[SAM15], [WYZ11], [XYZ17], [TBK+16], [BABM11], [FCH13], [WW13], [QLL+12], [CTA14], [LMZ09], [FMT+11], [SS09], [RP12], [AAA14], [WYZ10], [WYZS12], [XYZ19], [FBD+11], [LRRA11], [WC11], [FTLL14], [BMSZ09], [WKZ14], [LXLZ13], [AAA+15], [YVBG12], [Dam14],[WKO16], [SJ18], [XGZ07], [DG10], [ZAB15], [TJ14], [BABM09], [Bog18], [KZ13], [HZC11], [EZG15], [EZG14]

Table A.2: Thematical categorization 2/3, **microservices**, SOA, microservices and SOA

Maintainability Metrics and Prediction	**Evolution Management**	**Other**
[PR11], [SCKP08], [SP15], [DDD11], [RKS+10], [KRN+11], [AZD11], [KG12a], [PK16], [KG12b], [GL11], [KG12c], [BD13], [KSB+11], [Rom15], [BWZ17a], [PRF07], [dVE+13], [QX09], [PRFT07], [HO10], **[ELBH18]**, [GBO+10], [PK18], [SHM+13], [AZ11], [HZ12], [Wan18], [BFWZ18], [KKR17b], [WRC+13], [KAR+11], [SSP09], [GA11], [MM14], [Wan09], [MB10], [SVP12], [Hir12], [EM15], [TY13], [OSIS16], [Feu11], [ZHZ+16], [KR16], [CS15], [NW14], [HW08], [KKR17a], [PRT10], [BWZ17b], [Bog18], [NGS15], [WKO16], [KRS17]	[BKC+10], [KLS07], [Fun11], [WJ12], [IAKK16], [ZYL11], [XYCL17], [WW13], [MPS14], [LMZS09], [WC15], [ZSK12], [RP12], [WB14b], [WW15], [ZBA14a], [FSM14], [AP11], [IHL+13], [DMT13], [ZBA14b], [GMK+11], [SBB11], [Zuo16], [FL09], [ZGZW16], [SFCK14], [AP10], [AJP12], [Dem13], [TJSD10], [ZYA14], [PLW+10], [RRS+11], [MYXK08], [CCBJ11], [Jel15], [FS16], [HYX+11], [RCS+08], [FHPM11], [FS14a], [FHH+13], [SZZ+12], [KZ13], [WCX10], [HLM+11], [EZG15], [EZG14], [Fok15], [FS13a], [Fok14], [FOS15], [FS12], [FS14b], [FS13b]	[RGR11], [NM16], [WGEZ16], [SAD16], [KCH11], [BBPM15], [KLS08], [AD10], [ECZG12], [SMSL17], [LWD08], [LS], [ÖE12], [LS08], [DJC08], [AGR13], [WWR+12], **[TNSC17]**

Table A.3: Thematical categorization 3/3, **microservices**, SOA, microservices and SOA

B Full Methodical Categorization

Case Study, Field Study or Empirical Study	Literature Study
[HLK08], [KRN+11], [AZD11], [PK16], [KG12b], [XYZ19], [ZBA14a], [PRF07], [PRFT07], [IHL+13], [HO10], [MSK10], [GMK+11], [SBB11], [ECZG12], [Zuo16], [YVBG12], [Wan09], [ZGZW16], [SMSL17], [LWD08], [SFCK14], **[JTKB18]**, [AP10], [AJP12], [Dem13], [TJSD10], [ZYA14], [PLW+10], [RRS+11], [Ath17], [EKM07], [ÖE12], [MYXK08], [CCBJ11], [Feu11], [FS16], [DG10], **[SKH+17]**, [RCS+08], [AGR13], [LCZ12], [PMH+15], [FHH+13], [BABM09], [DCG11],[SZZ+12], **[GCF+17]**, [HZC11], [HLM+11], [SAM15], [PR11], [SCKP08], [SRY17], [SP15], [DDD11], [RKS+10], [XYCL17], **[MW18]**, [KS18], [FMT+11], [KSB+11], [Rom15], [RP12], [AS16], [PGM+15], [You16], [LM12], [WB14b], [LPM15], **[SAD16]**, [AZM+15], [dVE+13], [LRRA11], [EMK12], [PDMG14], [Pal13], [DMT13], [FTLL14], **[ZNL17]**, **[ELBH18]**, [GBO+10], [KLS08], [AD10], [SHM+13], [AZ11], [HZ12], [WKZ14], [LXLZ13], [NMV13], [Wan18], [PAK+14], [BFWZ18], [KKR17b], [WRC+13], [SSP09], [FL09], [GA11], **[KUSM18]**, **[CBD18]**, **[TL18]**, [Dam14], [Pis17], [WKO16], [OKIC17], [BCDP14], [Hir12], [TY13], [Jel15], [OSIS16], [PNM+13], [ZHZ+16], [MPN+12], [PMTG14], [XGZ07], [HYX+11], [HW08], [TSZ18], [KKR17a], [PRT10], [KD08], [FS14a], [NGS15], [KZ13], [REW+11], **[AAE18]**, [NBM+15], [WWR+12], [Pal15], **[TNSC17]**, [KRS17], [WCX10], [MZD11], [EZG15], [EZG14], [OGKI15], [DOK+17], [Fok15], [FS12], [FS14b], [FS13b]	[BB18a], [KG12c], [WW13], [BB18b], [CLW11], [NM16], [BD13], [SPR+19], [HPD+14], [BWZ17a], [AAA14], [SHM+13], [HZ12], [KKCR09a], [AAA+15], [WRC+13], [FL09], [MM14], [SVP12], [KKCR09b], [EM15], [KR16], [NW14], [SJ18], [Pal15]

Table B.1: Methodical categorization 1/2, **microservices**, SOA, microservices and SOA

Model or Taxonomy	Processes and Methods	Reference Architecture and Tools
[XYZ17], [SCKP08], [KLS07], [SP15], [Fun11], [KG12a], [IAKK16], [ZYL11], [GL11], [KG12c], [PM15], [BB18b], [QLL+12], **[MW18]**, [SS09], [WGEZ16], [WYZS12], [WW15], [EMK12], [AP11], [IHL+13], **[ZNL17]**, [GBO+10], [KLS08], [K C13], [Xa12], [MSK10], [AD10], [ZDZ10], [ZBA14b], [GMK+11], [SBB11], [Zuo16], [Lar08], [YVBG12], **[KUSM18]**, [ZGZW16], [SFCK14], [AP10], [AJP12], [Dem13], [TJSD10], [ZYA14], [PLW+10], [MB10], [RRS+11], [Hir12], [MYXK08], [FS16], [CS15], [NW14], [DG10], [HYX+11], **[SKH+17]**, [FHPM11], [ZAB15], [PRT10], [LCZ12], [BWZ17b], [PMH+15], [FHH+13], [BABM09], [NGS15], [SZZ+12], [KZ13], **[AAE18]**, **[GCF+17]**, [WCX10], [MZD11], [HLM+11], [Fok15], [FS13a], [FS12], [FS14b], [FS13b]	[XYZ17], [Koh08], [BKC+10], [LMN10], [HLK08], [SRY17], [Fun11], [IAKK16], [ZYL11], [BABM11], [XYCL17], [MPS14], **[MW18]**, [LMZ09], [LMZS09], [WC15], [ZSK12], [SS09], [KSB+11], [Rom15], [PGM+15], [You16], [LM12], [WB14b], [LPM15], [WYZ10], [WYZS12], [XYZ19], [FBD+11], [WW15], [ZBA14a], [FSM14], [AZM+15], [EMK12], [AP11], [IHL+13], [KCH11], [WC11], [PDMG14], [Pal13], [DMT13], [FTLL14], **[ZNL17]**, **[ELBH18]**, [GBO+10], [K C13], [Xa12], [MSK10], [ZDZ10], [KKCR09a], [NMV13], [GMK+11], [Wan18], [LEMP07], [Zuo16], [Lar08], [YVBG12], [FL09], **[KUSM18]**, [ZGZW16], [SFCK14], **[JTKB18]**, [AP10], [AJP12], [Dem13], [Dam14], [Pis17], [Jel15], [WKO16], [TJSD10], [ZYA14], [PLW+10], [SVP12], [OKIC17], [EKM07], [ÖE12], [BCDP14], [MYXK08], [TY13], [OSIS16], [PNM+13], [ZHZ+16], [MPN+12], [PMTG14], [XGZ07], [RCS+08], [TSZ18], [KD08], [TJ14], [LCZ12], [PMH+15], [BABM09], [Bog18], [DCG11], [SZZ+12], [KZ13], [NBM+15], **[GCF+17]**, [EZG11], [Pal15], [HZC11], [WCX10], [MZD11], [OGKI15], [DOK+17], [Fok15], [FS13a], [Fok14], [FS12], [FS14b]	[WYZ11], [RGR11], [SRY17], [WJ12], [TBK+16], [BWH18], [FCH13], [CTA14], **[MW18]**, [SS09], [KSB+11], [Rom15], [RP12], [PGM+15], [LM12], [WB14b], [LPM15], [WYZS12], [XYZ19], [BBPM15], [Pal13], [BMSZ09], **[ELBH18]**, [MSK10], [ZDZ10], [NMV13], [Wan18], [SBB11], [LEMP07], [WRC+13], [Zuo16], [YVBG12], **[KUSM18]**, [SMSL17], [LWD08], [Dem13], [FS13b], [TJSD10], [ZYA14], [PLW+10], [OKIC17], [RRS+11], [Ath17], [ÖE12], [BCDP14], [Hir12], [CCBJ11], [Jel15], [OSIS16], [PNM+13], [NPMG13], [ZHZ+16], [MPN+12], [PMTG14], **[SKH+17]**, [RCS+08], [AGR13], [FS14a], [NGS15], [DCG11], [REW+11], [NBM+15], **[GCF+17]**, [WWR+12], [Pal15], [HZC11], [MZD11], [HLM+11], [Fok15], [Fok14], [FOS15], [FS12]

Table B.2: Methodical categorization 2/2, **microservices**, SOA, microservices and SOA

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature