Research on Architecting Microservices: Trends, Focus, and Potential for Industrial Adoption

Paolo Di Francesco*, Patricia Lago†, Ivano Malavolta†
*Gran Sasso Science Institute, L'Aquila, Italy - paolo.difrancesco@gssi.it
†Vrije Universiteit Amsterdam, The Netherlands - {p.lago | i.malavolta}@vu.nl

***Abstract*—Microservices are a new trend rising fast from the enterprise world. Even though the design principles around microservices have been identified, it is difficult to have a clear view of existing research solutions for architecting microservices.**

In this paper we apply the systematic mapping study methodology to identify, classify, and evaluate the current state of the art on architecting microservices from the following three perspectives: publication trends, focus of research, and potential for industrial adoption. More specifically, we systematically define a classification framework for categorizing the research on architecting microservices and we rigorously apply it to the 71 selected studies. We synthesize the obtained data and produce a clear overview of the state of the art. This gives a solid basis to plan for future research and applications of architecting microservices.

***Index Terms*—Microservices, Software Architecture, Systematic Mapping Study**

I. Introduction

Netflix, Amazon, The Guardian and other companies have evolved their applications towards a microservice architecture (MSA). Lewis and Fowler define the microservice architectural style as *an approach for developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API* [4].

MSA arises from the broader area of Service Oriented Architecture (SOA) and focuses on specific aspects, such as componentization of small lightweight services, application of agile and DevOps practices for development, usage of infrastructure automation with continuous delivery features, decentralized data management and decentralized governance among services. There are many differences between SOA and MSA. For example, the design of services in MSA is driven by a share-nothing philosophy in order to support agile methods and promote isolation and autonomy. Instead, SOA adopts a share-as-much-as-you-can philosophy to promote a high degree of reuse [17]. Another significant difference is that MSA mainly focuses on service choreography, while SOA relies on both service orchestration and service choreography [17].

Even though the design principles around the microservice architectural style have been identified, many aspects are still unclear or unexplored. This makes it difficult for both researchers and practitioners to have a clear view of existing research solutions for architecting microservices, their characteristics, and their potential for broad industrial adoption. The **goal** of this paper is to characterize the current state of the art for understanding what we know about scientific research on architecting microservices.

For achieving this goal we applied the systematic mapping study **methodology**, which is a research methodology intended to provide an unbiased, objective and systematic instrument to answer a set of research questions by analysing all of the relevant research contributions in a specific research area. In our study we identified, classified, and evaluated the current state of the art on architecting microservices from different perspectives. We selected 71 primary studies from over three hundred potentially relevant papers; then, we rigorously defined a classification framework for precisely categorizing research results on architecting microservices, and we applied it to the 71 primary studies. Finally, we synthesized the obtained data to produce a clear overview of the state of the art in architecting microservices. Also, we assessed how research results on architecting microservices can be potentially transferred and adopted in industrial projects. This assessment can play the role of reference framework for acting towards a smoother transfer of research results to practice, which is one of the goals of an applied research field such as software engineering [5].

The main **contributions** of this study are: (i) a reusable framework for classifying, comparing, and evaluating architectural solutions, methods, and techniques (e.g., tactics, patterns, styles, views, models, reference architectures, or architectural languages) specific for microservices; (ii) an up-to-date map of the state of the art in architecting microservices and its implications for future research; (iii) an evaluation of the potential for industrial adoption of existing research results on architecting microservices;

The **audience** of this study is composed of both (i) *researchers* interested to further contribute to this research area, and (ii) *practitioners* interested to understand existing research on architecting microservices and thereby to critically adopt those solutions that best fit with their business goals.

The rest of the paper is organized as follows. In Section II we set the stage by giving the basic concepts around architecting microservices. The design of the study is presented in Section III, whereas its results are elaborated in Sections IV, V, and VI, where they are also put in a broader perspective and their potential implications for both researchers and practitioners are presented. Threats to validity and related work are described in Sections VII and VIII. With Section IX we close the paper and discuss future work.

II. Architecting Microservices

Common characteristics to the microservice architectural style are: (i) organization around business capability, (ii) automated deployment, (iii) intelligence in the endpoints, and (iv) decentralized control of languages and data. This style allows to design architectures that result flexible, modular and easy to evolve over time. Microservice architectures can provide significant benefits. Among the important ones, there is the possibility to design, develop, test and release services with great agility. Infrastructure automation allows to reduce the manual effort involved in building, deploying and operating microservices, thus enabling continuous delivery. Decentralized governance and data management allow services to be independent, and avoid an application to standardize on a single technology. Microservice architectures are particularly suitable for cloud infrastructures, as they greatly benefit from the elasticity and rapid provisioning of resources. Architecting microservices, however, is not an easy task as it requires to manage a distributed architecture and its challenges (e.g., network latency and unreliability, fault tolerance, data consistency and transaction management, communication layers, load balancing). Cloud infrastructures and new technologies play a fundamental role for realizing microservice architectures and managing the associated challenges and complexities.

III. Study Design

In this research we follow the well-established guidelines for systematic mapping studies [7, 16], in the following we present the key aspects of the design of our study.

A. Research Questions

We refined our research goal into three research questions:
*RQ1 – What are the **publication trends** of research studies about architecting microservices?* By answering this research question we aim at characterizing the intensity of scientific interest on architecting microservices, the relevant venues where academics are publishing their results on the topic, and their contribution types over the years.
*RQ2 – What is the **focus of research** on architecting microservices?* By answering this research question we aim at providing (i) a solid foundation for classifying existing (and future) research on architecting microservices and (ii) an understanding of current research gaps in the state of the art on architecting microservices.
*RQ3 – What is the **potential for industrial adoption** of existing research on architecting microservices?* By answering this research question we aim at assessing how and if the current state of the art on architecting microservices is ready to be transferred and adopted in industry.

B. Search and Selection Process

As shown in Figure 1, our search and selection process has been designed as a multi-stage process in order to have full control on the number and characteristics of the studies considered during the various stages.

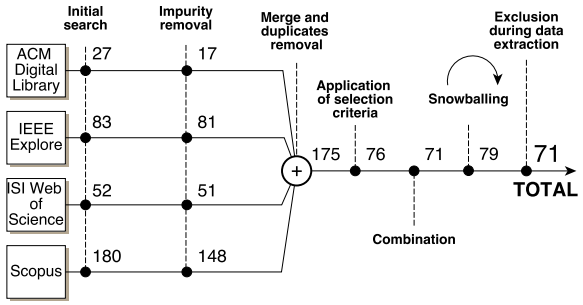

Fig. 1. Overview and numbers of the search and selection process

1. Initial search. As suggested in [7, 16], we performed automatic on four of the largest and most complete scientific databases and indexing systems in software engineering - ACM Digital Library, IEEE Xplore, Web of Science, and Scopus. The selection of these electronic databases and indexing systems is guided by: (i) the fact that they have been recognised as being an effective means to conduct systematic literature reviews in software engineering [16], (ii) their high accessibility, and (iii) their ability to export search results to well-defined, computation-amenable formats.

```
(architect* OR design* OR system OR structur*)
AND (microservi* OR micro-servi* OR "micro servi"*)
```

Listing 1. Search string used for automatic research studies

Our search string is shown Listing III-B, in order to cover as much relevant studies as possible we kept it very generic and considered exclusively the object of our research (i.e., existing research on architecting microservices). For consistency, the search string has been applied to title, abstract and keywords of papers in all the data sources considered in this research.
2. Impurity removal. Due to the nature of the involved data sources, search results included also elements that were clearly not research papers, such as international standards, textbooks, etc. In this stage we manually removed these results in order to have a coherent set of potentially relevant research studies.
3. Merging and duplicates removal. Here we combined all studies into a single dataset. Duplicated entries have been matched by title, authors, year, and venue of publication.
4. Application of selection criteria. We considered all the selected studies and filtered them according to a set of well-defined selection criteria. In this stage it was crucial to select studies objectively and in a cost-effective manner. To this purpose we used the adaptive reading depth [15], as the full-text reading of clearly excluded studies was not necessary. The inclusion and exclusion criteria of our study are:

I1 - Studies focussing on architectural solutions, methods or techniques (e.g., tactics, styles, reference architectures, or architectural languages) specific for microservices.
I2 - Studies providing an evaluation of the architectural solution, method or technique (e.g., via formal analysis, experiment, exploitation in industry, simple examples).
I3 - Studies subject to peer review.
I4 - Studies written in English.

E1 - Studies that, while focusing on microservices, do not explicitly deal with their architecture (e.g., studies focussing only on technological aspects, inner details of microservices).
E2 - Studies where microservices are only used as an example.
E3 - Secondary or tertiary studies (e.g., systematic literature reviews, surveys, etc.).
E4 - Studies in the form of tutorial papers, editorials, etc. because they do not provide enough information.
E5 - Studies not available as full-text.

5. Combination. If there were multiple papers on the same study, we kept a record of all of them and pointed them to a single study. This was necessary for ensuring completeness and traceability of our results [20].

6. Snowballing. We complemented the previously described automatic search with a snowballing activity [16]. The main goal of this stage is to enlarge the set of potentially relevant studies by considering each study selected in the previous stages, and focusing on those papers either citing and cited by it. More technically, we performed a closed recursive backward and forward snowballing activity [19].

C. Data Extraction

In this activity we (i) create a classification framework and (ii) collect data for each primary study. When going through the primary studies in detail for extracting information we agreed that 8 studies were semantically out of the scope of this research, so they have been excluded (see Figure 1). In order to have a rigorous data extraction process and to ease the management of the extracted data, we systematically designed a structured classification framework; it is composed of three facets, one for each research question of our study.

Publications trends (RQ1). The parameters we considered to collect data about publication trends are: publication year, publication venue (e.g., conference, journal, etc.), and research strategy (e.g., solution proposal, opinion paper, etc.).

Focus of research (RQ2). We followed a systematic process called *keywording* for defining the categories of this facet. Goal of the keywording process is to effectively develop a classification framework so that it fits the primary studies and takes their research focus into account [15]. The following details each step of the keywording process:

1. Identify starting set of studies. Two researchers randomly extracted 5 studies from the set of all primary studies; they have been used as pilot studies during the keywording process.

2. Identify keywords and concepts. Two researchers collected keywords and concepts by reading the full-text of each starting study. When all starting studies were analyzed, we combined all keywords and concepts to clearly identify the emerging context, nature, and contribution of the research on architecting microservices.

3. Cluster keywords and form categories. Two researchers performed a clustering operation on collected keywords and concepts in order to cluster them according to emerging categories. The output of this stage is the initial classification framework. Examples of emerging categories include: supported architecting activities, scope in the software lifecycle, considered quality attributes (e.g., reliability), etc. Next steps have been performed for each primary study.

4. Extract data from current study. A researcher extracted information about the current study to be analysed by collecting (i) information according to the parameters of the classification framework and (ii) any additional relevant information that did not fit within any parameter of the classification framework. If the collected information fit completely within the classification framework, then we proceeded to analyze the next study, otherwise the classification framework was refined.

5. Refine comparison framework. Two researchers discussed together on the collected additional information. This discussion could result either in the correction of the performed classification or in the refinement of the classification framework.

The above described process ended when there were no primary studies to analyze left. The specific parameters emerging from the keywording process are described in Section V.

Potential for industrial adoption (RQ3). This facet is composed of four different parameters: (i) readiness level for assessing the maturity of the involved technologies, (ii) industry involvement for understanding how academic and industrial researchers collaborate on the topic, (iii) tool support for distinguishing between software-based or knowledge-based contributions, and (iv) open-source test system for identifying existing benchmarks for microservice architectures.

D. Data Synthesis

The data synthesis activity involves collating and summarising the data extracted from the primary studies [8, § 6.5] with the main goal of understanding, analysing, and classifying current research on architecting microservices. Specifically, we performed a combination of content analysis (for categorizing and coding the studies under broad thematic categories) and narrative synthesis (for explaining in details and interpreting the findings coming from the content analysis).

E. Replicability of the Study

Due to page limitations we do not include all the details of the design of our study. To allow easy replication and verification of our study, a complete replication package[1] is publicly available to interested researchers. Our replication package includes: the detailed research protocol, the detailed description of all the parameters of the classification framework, the list of all selected studies, raw data for each phase of the study, and the R scripts for summarizing extracted data.

IV. Results - Publication Trends (RQ1)

Publication years. Figure 2 presents the distribution of publications on architecting microservices over the years. Here we have a clear confirmation of the scientific interest on architecting microservices in the last years. A small number of publications have been produced until 2014, which is actually the first year in which (i) microservices started to attract the

[1] http://cs.gssi.infn.it/ICSA2017ReplicationPackage

interest of large organizations, and (ii) the term microservice as architectural style was consistently used [14]. As a confirmation, even if the four studies published before 2014 were about systems composed of small-scale lightweight services (P10, P62, P63, P64), they were referring to slightly different perspectives on microservices as they are considered today. For example, P10 refers to low-level software components in the robotic domain as microservices, whereas P62 considers microservices as mobile services generated by end-users. We observed alternative definitions of microservices also in other cases throughout the years, as described in Section V-A.

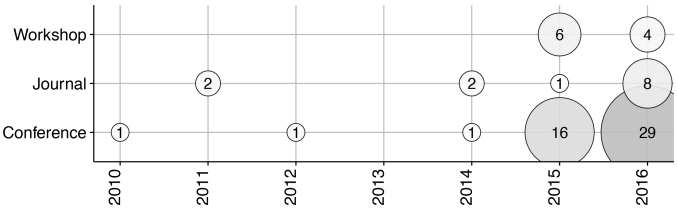

Fig. 2. Distribution of primary studies by type of publication over the years

Publication types. The most common publication type is *conference* papers (48/71), followed by *journal* (13/71) and *workshop* papers (10/71). Such a high number of conference and journal papers may indicate that architecting microservices is maturing as research topic despite its relative young age.

Publication venues. We can observe an extreme fragmentation in terms of publication venues, where research on architecting microservices is spread across 62 venues spanning different research areas like cloud infrastructures, software engineering, software services, autonomic computing, etc. This can be an indication that architecting microservices is considered as an orthogonal research target with many cross-cutting concerns.

Research strategies. Since this parameter is general and independent from the research area, we reuse the comparison of research approaches proposed by Wieringa et al. in [18]. We chose this comparison because (i) it has been widely used in various systematic mapping studies (e.g., in [3]) , and (ii) its categories are quite cost-effective to be identified [15].

As shown in Table I, here the clear winner is *solution proposal* (48/71). We can root this result to the fact that the microservice architectural style is still in its infancy (we recall here that the first well acknowledged definition has been provided only in 2014) and not yet consolidated in any (not even de facto) standards. This results in a large number of researchers trying to propose their own solutions for either recurrent or specific problems (see Section V-A for the details on this). *Validation research* (14/71) is the second most recurrent research strategy, highlighting the fact that researchers are actually providing some level of evidence about their proposed solutions, either by simulations, in-the-lab experiments, prototypes, etc. However, Table I also shows that researchers performed *evaluation research* very rarely (3/71), meaning that industry- and practitioners-oriented studies (e.g., industrial case study, action research, practitioner targeted survey) are not yet in the focus of researchers today. This represents a gap that should be filled by future research on architecting microservices, specially if we want to either (i) solve real problems coming from industrial scenarios or (ii) push the technology transfer of research results into industry.

TABLE I
APPLIED RESEARCH STRATEGIES

Res. strategies	#Studies	Studies
Solution proposal	48	P1, P2, P3, P4, P6, P7, P9, P10, P12, P13, P14, P15, P17, P18, P20, P21, P22, P23, P25, P26, P29, P30, P32, P33, P36, P39, P42, P43, P44, P45, P47, P49, P50, P51, P52, P54, P55, P56, P58, P61, P62, P63, P64, P66, P68, P69, P70, P71
Validation research	14	P3, P6, P9, P14, P21, P24, P38, P51, P52, P59, P61, P62, P69, P71
Opinion paper	8	P11, P16, P19, P28, P41, P48, P60, P67
Experience paper	8	P8, P31, P34, P37, P38, P40, P57, P65
Philosophical paper	3	P5, P46, P53
Evaluation research	3	P27, P35, P45

Main findings:

- Year 2015 signed a booming, monotonic increase in publication numbers with particular interest in conferences and journals (both increasing).
- The field is rooted in practice: publication venues are scattered across specific topics or application domains, and most publications propose specific solutions and validations thereof.

V. RESULTS - RESEARCH FOCUS (RQ2)

As described in Section III-C, the part of classification framework related to RQ2 has been systematically defined. After this process we obtained two main categories related to the research focus on architecting microservices, namely: scope (Section V-A) and support for architecting (Section V-B).

A. Scope

With this category we provide information to help researchers and practitioners in putting a research study on architecting microservices into context. For example, here we have parameters about the targeted problem, the main research contributions, the used definition of microservice, etc. In the following we discuss the results related to each parameter.

Target problems. As shown in Table II, the most recurrent problems targeted by the primary studies are *complexity* (19/71) and *low flexibility* (19/71), followed by *resources management* (16/71) and *service composition* (15/71). These results confirm that if on the one hand microservices can help in achieving a good level of flexibility (e.g., by promoting low services coupling, higher maintainability), on the other hand adopting a microservice-based architecture may bring higher complexity, mainly because they imply a high number of distributed services to operate. Interestingly, the bottom area of Table II shows problems that are related to system-level quality

TABLE II
TARGET PROBLEMS

Problems	#Studies	Studies
Complexity	19	P3, P10, P12, P18, P21, P22, P23, P26, P31, P33, P37, P39, P41, P46, P50, P62, P64, P67, P68
Low flexibility	19	P8, P15, P17, P19, P23, P25, P29, P37, P38, P47, P49, P50, P54, P61, P63, P66, P67, P68, P70
Resources management	16	P11, P15, P19, P21, P24, P27, P28, P34, P35, P36, P48, P49, P53, P54, P65, P70
Service composition	15	P2, P30, P32, P33, P37, P41, P47, P52, P55, P60, P66, P67, P68, P69, P70
Data management	13	P3, P4, P9, P19, P20, P29, P38, P54, P55, P61, P62, P63, P64
Modernization	11	P7, P8, P11, P13, P36, P42, P43, P51, P56, P65, P71
Low auditability	7	P6, P14, P21, P22, P37, P41, P59
Runtime uncertainty	7	P5, P10, P22, P26, P39, P40, P45
Low portability	5	P1, P12, P22, P46, P48
Low testability	5	P6, P22, P44, P57, P58
Realtime communication	4	P2, P25, P31, P37
Security	4	P14, P16, P28, P48
Time to market	4	P13, P47, P54, P66

like *low portability* and *testability* (5/71), *security* and *time to market* (4/71). These have been extensively investigated in the software architecture area, but are still new to microservice architectures; this result may be an indicator of a potentially relevant research gap needing attention in the future.

Research contribution. The main research contributions are *application* (21/71) and *method* (18/71), as reported in Table III. The high number of *applications* might indicate that microservice architerectures can be suitable in many areas, and it would confirm that the trend emerged from the industrial field. Researchers seem to deal with the complexities in architecting microservices by proposing new *methods* (18/71), *reference architectures* (16/71) and *middlewares* (12/71). A significant number of *problem framing* (14/71) studies indicates that researchers are trying to better undestand and shape the challenges in this field. Interestingly, few papers are investigating *design patterns* and *architectural languages* for microservices, which might reveal gaps to be filled by the research community. Proposing an *architectural language* for microservices could help architects in many activities as for example in reasoning on the system as a whole, performing analysis on the system qualities, coping with the dynamic and changing aspects of the application at runtime.

Research perspective. The research perspective is the main focus area of the primary study. As reported in Table IV, the predominant research perspectives are: *cloud* (21/71), *system quality* (21/71) and *migration* (16/71). The attention on *cloud* confirms the close relation between the microservice architectural style with the DevOps culture, and also confirms that containerization and virtualization are key enabling technologies. Moreover, the focus on the *system quality* (e.g., scalability, performance, security) suggests that the microservice architectural style has direct impact on the design of a system

TABLE III
RESEARCH CONTRIBUTION

Contribution	#Studies	Studies
Application	21	P3, P8, P9, P10, P12, P15, P17, P20, P21, P25, P29, P31, P36, P37, P40, P47, P53, P54, P59, P65, P66
Method	18	P7, P22, P24, P26, P27, P35, P42, P43, P51, P56, P57, P58, P61, P62, P64, P69, P70, P71
Reference architecture	16	P1, P2, P3, P5, P13, P18, P21, P33, P34, P38, P46, P49, P55, P59, P60, P63
Problem framing	14	P11, P12, P16, P19, P28, P30, P32, P33, P39, P41, P46, P48, P57, P67
Middleware	12	P3, P4, P6, P14, P23, P32, P40, P44, P45, P61, P62, P68
Design pattern	2	P30, P50
Architectural language	2	P52, P62

and that researchers are still investigating how to leverage its characteristics. Not surprisingly, a significant number of studies are investigating *migration* techniques in order to adopt and benefit of microservices starting from the so called monolithic applications. It is also interesting to note that the domains of application of microservices are rather fragmented. Indeed, the microservice architecture have been applied to recent technologies as *Internet of Things* (6/71) and *mobile* (6/71), but also to *domain-specific* (11/71) fields as robotics (P10) or data centers (P29).

TABLE IV
RESEARCH PERSPECTIVE

Perspective	#Studies	Studies
Cloud	21	P1, P11, P12, P16, P17, P21, P24, P27, P28, P32, P34, P35, P36, P41, P42, P46, P48, P53, P58, P65, P70
System quality	21	P6, P7, P14, P16, P20, P22, P23, P24, P27, P28, P34, P35, P39, P44, P45, P49, P50, P57, P58, P64, P66
Migration	16	P7, P8, P11, P36, P37, P39, P42, P43, P48, P51, P53, P56, P63, P65, P67, P71
Domain-specific	11	P3, P9, P10, P15, P18, P25, P26, P29, P31, P37, P69
IoT	6	P13, P23, P30, P40, P47, P54
Mobile oriented	6	P54, P55, P61, P62, P64, P66
Other	10	P2, P4, P5, P19, P33, P38, P52, P59, P60, P68

Vertical scope. It is the layer of abstraction at which the study considers microservice architectures. As shown in Table V, more than half of the studies focus on the *service* layer only (39/71) without considering other layers. Differently, other studies not only focus on services, but also consider the environment upon which the services run. Indeed, the *container* (14/71) layer and the *virtual machine* (13/71) layer are discussed as relevant aspects of the architectures.

Software lifecycle scope. Results show that when architecting microservices, the predominant software lifecycle phases are: *design* (65/71), *implementation* (24/71) and *operation* (22/71) – see Table VI. Surprisingly, there is a significant gap between the number of studies on the *design* phase and both *implementation* and *operation*. This gap might confirm the

TABLE V
VERTICAL SCOPE

Vertical scope	#Studies	Studies
Service	39	P1, P3, P4, P5, P6, P7, P9, P10, P13, P15, P18, P21, P22, P23, P33, P37, P38, P39, P40, P42, P44, P45, P50, P51, P52, P54, P55, P56, P57, P60, P61, P62, P63, P64, P66, P67, P69, P70, P71
Container	14	P2, P8, P11, P17, P19, P25, P29, P35, P36, P47, P48, P58, P59, P68
Virtual machine	13	P12, P14, P16, P20, P24, P26, P30, P31, P32, P34, P43, P49, P53
Hardware	4	P27, P28, P41, P46
Operating system	1	P65

existence of challenges and complexities in the implementation and deployment of microservice architectures in practice. It is worth noting that the phases of *maintenance* (12/71) and *testing* (10/71) are not a primary target of research, probably because microservice applications are relatively young. Possibly, the areas of microservices *maintenance* and *testing* might become fields for further investigations. We observe also that in only (1/71) study (P8) the *requirements* are discussed in detail along with their impact on the evolution of the presented microservice-based system.

TABLE VI
SOFTWARE LIFECYCLE SCOPE

Lifecycle scope	#Studies	Studies
Design	65	P1, P2, P3, P4, P5, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19, P20, P21, P23, P24, P25, P26, P27, P28, P29, P30, P31, P32, P33, P34, P35, P36, P37, P38, P39, P40, P41, P42, P43, P46, P47, P48, P49, P50, P51, P52, P53, P54, P55, P56, P59, P60, P61, P62, P63, P64, P65, P66, P67, P68, P69, P70, P71
Implementation	24	P8, P9, P10, P14, P15, P20, P21, P25, P29, P30, P31, P34, P35, P37, P38, P52, P54, P56, P60, P61, P64, P66, P68, P70
Operation	22	P1, P8, P11, P12, P16, P17, P19, P24, P25, P26, P27, P30, P32, P35, P36, P41, P43, P48, P53, P59, P62, P65
Maintenance	12	P7, P30, P31, P37, P43, P51, P54, P56, P63, P68, P70, P71
Testing	10	P6, P17, P22, P25, P43, P44, P45, P56, P57, P58
Requirements	1	P8

Microservice architecture definition. In the primary studies microservice architectures have been defined in several ways and in some cases even more than one single definition was reported. The most recurring definition was the one provided by *J. Lewis and M. Fowler* (32/71) [4], followed by the one given by *S. Newman* (13/71) [13]. In (16/71) studies the provenance of the definitions was scattered among *other* scientific papers, while in (19/71) studies *own-informal* definitions were adopted. This evident fragmentation of definitions results in ambiguities and does not help to provide clarity to the boundaries of the microservice architectural style. Interestingly, the definitions provided by *J. Lewis and M. Fowler* and *S. Newman* (13/71) seem to start prevailing over the remaining definitions.

B. Support for architecting

Here we characterize research studies with respect to how they support architecture-specific concerns and activities.

Architecting activities. We have based our classification of architecting activities according to the introvert/extrovert nature of software architects [11]. The introvert nature is composed of the analysis and other design-oriented architectural activities and it has been refined into the architecting activities defined by Li et al. in [10]. The extrovert nature regards the communication between architects and other stakeholders and it has been further classified into the *providing information* and *getting input* parameters proposed by Kruchten in [9]. As shown in Table VII, the mainly discussed introvert architecting activities are *analysis* (56/71) *implementation* (29/71), *description* (23/71) and *evaluation* (18/71). These indicators seem to show that researchers are not focusing on architecting activities regarding *architectural reuse* (6/71) or *maintenance and evolution* (15/71) of existing assets. Moreover, *architecture impact analysis* (2/71) and *architecture recovery* (5/71) are hardly discussed. These aspects may be significant research directions to explore. In the lower part of the Table we can observe how little investigation is performed on extrovert architecting activities, i.e., *providing information* (4/71) and *getting input* (0/71). These complementary activities to the system design concern the interaction with other stakeholders. From a research perspective the low interest in these complementary activities might indicate that there could be areas of improvement in the engagement of customers and users, and also in the project management and the communication with the teams. Toward these directions an architectural language might be a powerful instrument in order to enable better communication with both stakeholders and developers.

Quality attributes. It is the set of quality attributes addressed or discussed by the study. *Performance efficiency* (40/71), and *maintainability* (28/71) are the most investigated quality attributes, while the remaining qualities are almost equally represented, as shown in Table VIII. One of the key reasons why *performance efficiency* is extensively investigated is because it includes scalability, which microservice architectures aim to support to a great extend. Similarly, the attention to *maintainability* might be related to the characteristics of small services and automatic deployment. We observe that, the difference between the higher focus on the *maintainability* quality attribute with respect to the activity of *maintenance and evolution* (discussed in the previous paragraph) is due to the fact that researchers address more often the quality of the system rather than performing the activity of maintenance as defined by Li et al. [10] as the activity of *correcting faults and adapting to a changed or changing operational environment*. If we compare the number of primary studies addressing *security* (17/71), *reliability* (14/71) and *portability* (12/71) to the number of studies focusing on *performance efficiency* (40/71) we might observe that these quality attributes have not received the same research attention. Possible research gaps might exist in the areas related to these quality attributes.

TABLE VII
ARCHITECTING ACTIVITIES

Arch. activities	#Studies	Studies
Introvert activities		
Architectural Analysis	56	P1, P2, P3, P4, P5, P7, P8, P9, P10, P11, P12, P13, P14, P15, P16, P17, P18, P19, P20, P21, P23, P25, P26, P28, P29, P31, P32, P33, P34, P35, P36, P37, P38, P39, P40, P42, P45, P47, P48, P49, P50, P51, P54, P55, P56, P57, P60, P61, P63, P64, P65, P66, P67, P68, P69, P71
Architectural Implementation	29	P2, P3, P8, P10, P11, P14, P15, P18, P20, P21, P25, P26, P29, P32, P33, P34, P37, P38, P47, P49, P52, P54, P56, P60, P61, P66, P68, P69, P70
Architecture Description	23	P1, P3, P7, P8, P18, P37, P38, P40, P41, P51, P52, P53, P55, P56, P59, P61, P62, P65, P66, P68, P69, P70, P71
Architectural Evaluation	18	P8, P12, P14, P15, P24, P25, P26, P29, P32, P35, P37, P49, P51, P55, P56, P61, P64, P65
Architectural Maintenance and Evolution	15	P1, P4, P6, P7, P8, P31, P37, P41, P43, P44, P45, P51, P53, P58, P70
Architecture Understanding	10	P1, P7, P8, P11, P12, P26, P31, P37, P45, P50
Architecture Reuse	6	P8, P11, P30, P37, P67, P70
Architectural Synthesis	6	P8, P12, P13, P37, P51, P62
Architecture Recovery	5	P9, P15, P27, P43, P65
Architecture Impact Analysis	2	P7, P8
Extrovert activities		
Providing Information	4	P22, P40, P46, P70
Getting Input	0	

TABLE VIII
QUALITY ATTRIBUTES

Quality attr.	#Studies	Studies
Performance efficiency	40	P1, P3, P4, P6, P7, P9, P11, P12, P14, P15, P17, P18, P19, P20, P23, P24, P25, P26, P27, P29, P30, P31, P32, P33, P34, P35, P36, P41, P49, P51, P52, P54, P57, P58, P61, P65, P66, P67, P68, P69
Maintainability	28	P1, P3, P4, P6, P8, P11, P14, P17, P22, P23, P30, P33, P36, P37, P38, P39, P40, P43, P45, P47, P50, P51, P53, P56, P63, P65, P68, P71
Security	17	P1, P11, P14, P16, P17, P18, P19, P20, P21, P28, P32, P52, P54, P58, P60, P64, P66
Functional suitability	14	P2, P6, P11, P12, P29, P30, P38, P42, P47, P50, P55, P56, P59, P62
Reliability	14	P1, P5, P6, P12, P30, P32, P36, P44, P45, P49, P54, P57, P65, P68
Compatibility	14	P4, P5, P10, P18, P19, P21, P23, P30, P40, P53, P56, P58, P60, P68
Usability	13	P8, P9, P20, P25, P29, P33, P34, P37, P51, P52, P56, P61, P62
Portability	12	P1, P11, P12, P13, P17, P30, P46, P48, P53, P56, P65, P70

TABLE IX
ARCHITECTURE PROVENANCE

Arch. provenance	#Studies	Studies
Designed	58	P1, P2, P4, P5, P7, P10, P11, P12, P13, P14, P16, P18, P19, P20, P21, P22, P23, P24, P25, P26, P28, P29, P30, P32, P33, P34, P35, P36, P38, P39, P40, P41, P42, P43, P46, P47, P48, P49, P50, P51, P52, P53, P55, P56, P57, P58, P59, P60, P61, P62, P63, P64, P66, P67, P68, P69, P70, P71
Extracted	13	P3, P6, P8, P9, P15, P17, P27, P31, P37, P44, P45, P54, P65

Architecture provenance. An architecture is *designed* if it is created prior its implementation, *extracted* vice versa. The type of provenance of the architectures results to be mainly *designed* (58/71) rather than *extracted* (13/71), as reported in Table IX. This result might suggest that it is not easy to realize microservice architectures unless an actual analysis and design of the system has been performed. From a research point of view, this might confirm that there exist some uncertainty about the realization of microservices. This uncertainty might be related either to the microservice architectural style challenges, or the needed infrastructure and technologies.

Architectural language. From the analysis of the primary studies has emerged that the majority of the proposed architectures were described using *informal architectural languages* (50/71) while in few cases *UML* (4/71) was used. Interestingly, six different architectural languages were either *used* or proposed as suitable languages for designing microservice architectures: *UML* (P43), *BPMN* (P33, P51), *Medley* (P52), *OCCIEx* (P59), *Ciudad* (P62) and *Diary* (P70). From a researcher's point of view, the use of *informal architectural languages* and the lack of a predominant architectural language can indicate difficulties in the description and modeling of microservice architectures.

Architecture description types. The results show that the architectures proposed were mostly described in their *structural* (55/71) aspects, while the *behavioral* (24/71) aspects were treated in a minor number of cases. In the remaining studies (12/71) this classification was not applicable.

Design patterns. Several design patterns have being discussed in the primary studies, but not many have been applied frequently. The most recurring design patterns are: *API gateway* (11/71), *publish/subscribe* (8/71), *circuit breaker* (6/71), *proxy* (4/71) and *load balancer* (3/71). It is important to note that a number of other design patterns have been proposed, and as the field of microservice architectures matures more design patterns will likely emerge. Results are reported in Table X.

Technology-specific. We classified as *technology-specific* the studies which have proposed solutions, methods or techniques that are dependent from one or more specific technologies. The results have shown that (54/71) of the primary studies were *not technology-specific*, while the remaining (17/71) resulted to be *technology-specific*. The predominance of *not technology-specific* studies is a good indicator because approaches and solutions can be reused. Differently, *technology-specific* studies have the advantage of being more detailed, but their applicability and portability in the future might be limited.

TABLE X
DESIGN PATTERNS

Design patterns	#Studies	Studies
API Gateway	11	P2, P3, P8, P18, P33, P35, P38,P47, P50, P65, P67,P68, P71
Publish/subscribe	8	P2, P3, P4, P19, P40, P50, P67, P68
Circuit breaker	6	P1, P16, P30, P44, P46, P65
Proxy	4	P6, P8, P12, P18
Load balancer	3	P30, P47, P49
Other	9	P1, P11, P19, P30, P38, P44,P46,P50, P67

Infrastructure services. It is the set of infrastructure services supporting non-functional tasks, as defined by Richards in [17]. As shown in Table XI, the most relevant infrastructure services seem to be related to *monitoring* (26/71) and *system level management* (26/71). Microservice architectures, being inherently distributed, show a clear need for *monitoring* capabilities (e.g., logging, profiling) but also need instruments for *system level management* (e.g., health management, autoscaling, load balancing) in order to leverage the underlying infrastructure efficiently. Once more this confirms the tight coupling between microservices and DevOps.

TABLE XI
INFRASTRUCTURE SERVICES

Infr. services	#Studies	Studies
Monitoring	26	P1, P9, P10, P14, P15, P16, P18, P20, P26, P27, P28, P29, P30, P31, P32, P34, P37, P38, P39, P41, P50, P53, P59, P60, P61, P65
System level management	26	P1, P2, P11, P12, P15, P18, P20, P25, P26, P34, P35, P36, P37, P40, P41, P43, P45, P46, P47, P48, P49, P50, P55, P59, P62, P68
Service Orchestration	13	P1, P3, P6, P8, P12, P15, P20, P46, P48, P49, P66, P68, P69
Service brokering	10	P1, P11, P12, P20, P29, P31, P32, P40, P50, P68
Messaging	6	P9, P21, P40, P53, P54, P68
Security	6	P14, P20, P21, P23, P28, P62
Service Proxies	4	P6, P12, P44, P67
Data storage	1	P18

Not surprisingly, a significant research interest is pointing to *service orchestration* (13/71) and *service brokering* (10/71), confirming that service management capabilities are fundamental to this area. Interestingly, some research is also focused on providing *security* (6/71) at the infrastructural layer.

Main findings:

- ▶ Both research scope and support for architecting mirror the current immaturity of the field.
- ▶ In *scope* are problems that consolidate the need to master the tradeoffs between complexity and flexibility; focus on cloud and mobile paradigms (and many domains that rely on them); and the focus on legacy migration. Also, most studies cover the design phase, and only one does address requirements – food for thought.
- ▶ Architecture analysis emerges as the most popular *architecting activity*. Results suggest software architecture as a powerful instrument for stakeholder engagement. The clear focus on infrastructure services will help devising new patterns and styles building upon them and hence further leveraging cloud-based architecture models.

VI. RESULTS - POTENTIAL FOR INDUSTRIAL ADOPTION (RQ3)

Readiness level. Defined by the systematic measurement system for assessing the maturity of a particular technology [12], the technology readiness level (TRL) is an integer n where $1 \leq n \leq 9$. This measure has been proposed by the Horizon 2020 European Commission for the 2014/2015 work program.

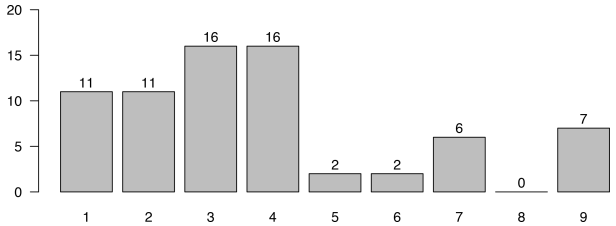

Fig. 3. Frequencies of technology readiness levels

Figure 3 presents the frequencies of the TRLs of our primary studies. Firstly, the majority of the studies (54/71) have a low TRL (i.e., TRL $\leq$ 4), where a technology is either formulated, validated or demonstrated at most in lab. Secondly, 4 studies have a medium TRL (i.e., 5 $\leq$ TRL $\leq$ 6), where a technology is either validated or demonstrated in the relevant environment, and 13 studies have a high TRL (i.e., TRL $\geq$ 7), where the technology is either completed, demonstrated, or proven in operational environment. These results indicate that (i) research on architecting microservices is still in its initial phases for what concerns the transferability of the developed technologies into industry (many of them are simply not ready yet), (ii) in many studies (22/71) only the basic principles on architecting microservices have been discussed (TRL=1) or the technology concept has been simply formulated (TRL=2)and (iii) there is a relatively large number of studies in which the actual system has been proven in its operational environment (P4, P8, P20, P34, P37, P61, P68). Interestingly, the studies with TRL=9 are evenly distributed between academic only, industrial only, and mixed involvement (see next parameter).

Industry involvement. Here we classify each primary study as: *academic* if all authors are affiliated with universities or research centers, *industrial* if all authors are affiliated with some companies, or a *mix* of the previous two categories. As shown in Figure 4, the majority of primary studies contribute with an academic-only perspective (41/71), followed by mixed (18/71) and industry-only (12/71) contributions. This result is encouraging as in almost half of the primary studies there is the involvement of at least one industrial researcher or

practitioner; this means that in those cases we are potentially smoothing the knowledge exchange between academia and industry, where research is performed on industrially relevant problems and new methods, technologies and tools are transferred from academia to industry [20].

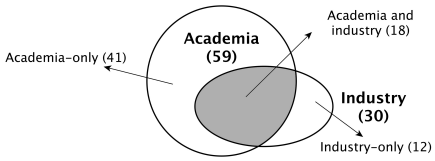

Fig. 4. Distribution of industry involvement

Tool support. In the context of this study a tool can be considered as an *instance that may represent a precise version of an automated tool or a written procedure* [6]. Based on the given definition, we categorize a tool either as *software based* or *knowledge based*. Our analysis shows that almost half of the studies (35/71) provide a knowledge-based tool (e.g., best practices, documented design patterns, guidelines), whereas only 20 studies provide a software-based tool (e.g., a testing tool, a code generator, a monitoring infrastructure) and 16 studies provide a combination of knowledge- and software-based tool (e.g., a method for implementing services that communicate via an implemented middleware, a testing tool together with a testing method, etc.).

Open-source test system. When screening the 71 primary studies we checked if an open-source test system for benchmarking microservices-based systems was used, discussed or proposed. We identified only one such systems (in P27), it is called Acme Air and it is publicly available as open-source repository on GitHub[2]. Acme Air is a web-based system available in two different architectures (i.e., monolithic service and microservice) and in two different languages (i.e., Node.js and Java), thus providing to researchers a very useful benchmark for evaluating, measuring, and comparing their own solutions over a common reference system.

Main findings:

- In spite of their focus on specific solutions, the low TRL scores of most studies suggest that industrial transferability is far away.
- The balanced involvement of industrial and academic authors is however promising for knowledge co-creation and cross-fertilization.

VII. Threats to Validity

In 2015, Petersen et al. [16] created a checklist for objectively assessing the quality of systematic mapping studies. In this context a score can be computed as the ratio of the number of actions taken in a study in comparison to the total number of actions in the checklist. In our case we achieve a score of 65%, far higher than most systematic studies in the literature, which have a distribution with a median of 33% and 48% as the absolute maximum value. As always, however, threats to validity are unavoidable. The following reports on the main threats to validity to our study and how we mitigated them.

External validity. The most severe external threat of our study consists in the fact that our primary studies are not representative of the state of the art on architecting microservices. As a solution, we applied a search strategy consisting of both automatic search and backward-forward snowballing on the selected studies in combination. Also, we considered only peer-reviewed papers and excluded the so-called grey literature (e.g., white papers, editorials, etc.); nevertheless, this potential bias did not impact our study significantly since considered papers have undergone a rigorous peer-reviewed process, which is a well-established requirement for high quality publications. We also applied well-defined and previously validated inclusion and exclusion criteria, which we refined iteratively by considering the pilot studies of our review.

Internal validity. We rigorously defined the research protocol of our study and we iteratively defined our classification framework by rigorously applying the keywording process. The syntheses of the collected data have been performed by applying well-assessed descriptive statistics. During the horizontal analysis we made a sanity test of the extracted data by cross-analyzing parameters of the classification framework.

Construct validity. We mitigated this potential bias by automatically searching the studies on multiple data sources, independently of publishers' policies or business concerns; also we are reasonably confident about the construction of the search string since the terms used are very general and suited to our research questions; the automatic search has been complemented with snowballing. Also, we rigorously selected the potentially relevant studies according to well-documented inclusion and exclusion criteria. This selection stage was performed by one researcher and, as suggested by [20], a random sample of potentially relevant studies was identified and the inter-researcher agreement was ensured.

Conclusion validity. We rigorously defined and iteratively refined our classification framework, so that we could reduce potential biases during the data extraction process. In so doing we also have the guarantee that the data extraction process was aligned with our research questions. More in general, we mitigated potential threats to conclusion validity by applying the best practices coming from three different guidelines on systematic studies [7, 16, 20]. We applied those best practices in each phase of our study and we documented each phase in a publicly available research protocol, thus making our study easy to be replicated by other researchers.

VIII. Related Work

A systematic mapping on microservices was performed by Pahl et al. on a set of 21 primary studies from 2014 to 2015 [14]. It is a classification of the research directions in the field and highlights the relevant perspectives considered by researchers. Our study differs from [14] in the following terms: (i) we apply a more comprehensive search process by considering studies published in any year up to 2016 (allegedly the term *microservices* has been used for the first time in 2011

[2]https://github.com/acmeair/acmeair

[http://en.wikipedia.org/wiki/Microservices]), extending their search string, and complementing the automated search with snowballing; (ii) we apply a systematic process for defining a classification framework; (iii) we investigate on the potential of industrial adoption of research in architecting microservices.

In [1] Alshuqayran, Ali and Evans presented a systematic mapping study on microservice architecture. Their study focusses on (i) the architectural challenges faced by microservice-based systems, (ii) the architectural diagrams used for representing them, and (iii) the involved quality requirements. The work by Alshuqayran et al. and our study can be considered as complementary, each of them cutting the topic of architecting microservices from different perspectives. The main difference between those two studies is that ours considers different research questions, thus leading to different results, findings, and overlook for future research.

Dragoni et al. performed an informal survey on microservices [2]. Our study differs from their study because (i) we specifically focus on architectural principles, method, and techniques, rather than on microservices in general; (ii) we apply a rigorous empirical method throughout the study (i.e., systematic mapping), thus providing evidence-based results and easing replication of the performed research; (iii) the objective of our study is to characterize existing research on architecting microservices, rather than on providing a narrative viewpoint on their historical, current, and future traits.

IX. Conclusions and Future Work

The purpose of this study is to provide *a broader survey investigating relationships among research contributions on microservices is demanded* [2]. Specifically, we performed a systematic mapping on 71 primary studies and produced a clear overview of the state of the art on architecting microservices. The results of this study will benefit both researchers willing to further contribute to the area and practitioners willing to understand existing research. *Future work* includes (i) a qualitative study involving practitioners aiming at better understanding the state of the practice on microservices and (ii) attacking a selection of the identified research gaps.

References

[1] N. Alshuqayran, N. Ali, and R. Evans. A Systematic Mapping Study in Microservice Architecture. In *Proc. of the 9th International Conference on Service-Oriented Computing and Applications*. IEEE, IEEE, 2016.

[2] N. Dragoni, S. Giallorenzo, A. L. Lafuente, M. Mazzara, F. Montesi, R. Mustafin, and L. Safina. Microservices: yesterday, today, and tomorrow. *arXiv preprint arXiv:1606.04036*, 2016.

[3] E. Engström and P. Runeson. Software product line testing - a systematic mapping study. *Inf. Softw. Technol.*, 53(1):2–13, Jan. 2011.

[4] M. Fowler and J. Lewis. Microservices a definition of this new architectural term. *URL: http://martinfowler.com/articles/microservices.html*.

[5] M. Ivarsson and T. Gorschek. A method for evaluating rigor and industrial relevance of technology evaluations. *Empirical Software Engineering*, 16(3):365–395, 2011.

[6] M. L. Jaccheri, G. P. Picco, and P. Lago. Eliciting software process models with the e 3 language. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 7(4):368–410, 1998.

[7] B. Kitchenham and P. Brereton. A systematic review of systematic review process research in software engineering. *Information and software technology*, 55(12):2049–2075, 2013.

[8] B. A. Kitchenham and S. Charters. Guidelines for performing systematic literature reviews in software engineering. Technical Report EBSE-2007-01, Keele University and University of Durham, 2007.

[9] P. Kruchten. What do software architects really do? *Journal of Systems and Software*, 81(12), 2008.

[10] Z. Li, P. Liang, and P. Avgeriou. Application of knowledge-based approaches in software architecture: A systematic mapping study. *Information and Software Technology*, 55(5):777–794, 2013.

[11] I. Malavolta, P. Lago, H. Muccini, P. Pelliccione, and A. Tang. What industry needs from architectural languages: A survey. *IEEE Transactions on Software Engineering*, 39(6):869–891, 2013.

[12] J. C. Mankins. Technology readiness levels. *White Paper, April*, 6, 1995.

[13] S. Newman. *Building Microservices*. O'Reilly Media, Inc., 2015.

[14] C. Pahl and P. Jamshidi. Microservices: A Systematic Mapping Study. In *Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy*, pages 137–146, 2016.

[15] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. Systematic mapping studies in software engineering. In *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering*, EASE'08, pages 68–77, Swinton, UK, UK, 2008.

[16] K. Petersen, S. Vakkalanka, and L. Kuzniarz. Guidelines for conducting systematic mapping studies in software engineering: An update. *Information and Software Technology*, 64:1–18, 2015.

[17] M. Richards. *Microservices vs. Service-Oriented Architecture*. O'Reilly Media, 2015.

[18] R. Wieringa, N. Maiden, N. Mead, and C. Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements Engineering*, 11(1):102–107, 2006.

[19] C. Wohlin. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, pages 38:1–38:10, New York, NY, USA, 2014. ACM.

[20] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering*. Computer Science. Springer, 2012.