Architecting with microservices: A systematic mapping study

Paolo Di Francesco\textsuperscript{a,\ast}, Patricia Lago\textsuperscript{b}, Ivano Malavolta\textsuperscript{b}

\textsuperscript{a} Gran Sasso Science Institute, L’Aquila, Italy
\textsuperscript{b} Vrije Universiteit Amsterdam, the Netherlands

\begin{abstract}
Context: A microservice architecture is composed of a set of small services, each running in its own process and communicating with lightweight mechanisms. Many aspects on architecting with microservices are still unexplored and existing research is still far from being crispy clear.

Objective: We aim at identifying, classifying, and evaluating the state of the art on architecting with microservices from the following perspectives: publication trends, focus of research, and potential for industrial adoption.

Method: We apply the systematic mapping methodology. We rigorously selected 103 primary studies and we defined and applied a classification framework to them for extracting key information for subsequent analysis. We synthesized the obtained data and produced a clear overview of the state of the art.

Results: This work contributes with (i) a classification framework for research studies on architecting with microservices, (ii) a systematic map of current research of the field, (iii) an evaluation of the potential for industrial adoption of research results, and (iv) a discussion of emerging findings and implications for future research.

Conclusion: This study provides a solid, rigorous, and replicable picture of the state of the art on architecting with microservices. Its results can benefit both researchers and practitioners of the field.

© 2019 Elsevier Inc. All rights reserved.
\end{abstract}

1. Introduction

Amazon, Netflix, LinkedIn, Spotify, SoundCloud and other companies (Fowler and Lewis, 2014; Villamizar et al., 2015; Yahia et al., 2016) have evolved their applications towards a microservice architecture (MSA). The most acknowledged definition of the microservices architectural style is the one provided by Fowler and Lewis (2014), which describes it 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.

Recently, the microservice architectural style has received significant attention from a research point of view. However, as of today it is difficult for both researchers and practitioners to have a clear view of existing research solutions for architecting with microservices. The goal of this paper is to characterize the current state of the art for understanding what we know about scientific research on architecting with microservices.

For achieving this goal we designed and conducted a systematic mapping study methodology. Specifically, we select 103 primary studies from 532 potentially relevant papers, we rigorously define a classification framework for categorizing research results on architecting with microservices, and we apply it to the 103 primary studies. Finally, we synthesize the obtained data to produce a clear overview of the state of the art in architecting with microservices. Also, we assess how research results on architecting with 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.

The main contributions of this study include:

\begin{itemize}
\item 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;
\item an up-to-date map of the state of the art in architecting with microservices;
\item an evaluation of the potential for industrial adoption of existing research results on architecting with microservices;
\item an evidence-based discussion of the emerging research trends, patterns, and gaps, and their implications for future research on architecting with microservices.
\end{itemize}
This study is an extended version of our previous research on architecting with microservices (Di Francesco et al., 2017b). The novelties added in this study are: (i) the extension of the set of primary studies from 71 to 103 entries because now we cover publications until the beginning of May 2017, (ii) a more in-depth elaboration of the extracted data, (iii) an orthogonal analysis about the potential interactions between various parameters of the classification framework, (iv) the analysis of the research trends over the years.

The audience of this study is composed of both (i) researchers interested to investigate on the microservices architectural style, and (ii) practitioners willing to understand and adopt existing research on architecting with microservices.

The remainder of this paper is organized as follows. In Section 2 we provide basic concepts about architecting with microservices. In Section 3 we present the design of the study. The elaborated results are reported in Sections 4, 5, and 6. Section 7 discusses the orthogonal findings of the study. Threats to validity and related work are described in Sections 8 and 9, respectively. Section 10 closes the paper.

2. Architecting with microservices

While there has not been a wide acceptance of a specific definition, a popular one was provided by Lewis and Fowler, which define the microservice architectural style as an approach to 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 (Fowler and Lewis, 2014). Recurrent characteristics of the microservice architectural style are: (i) organization of the system 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 should be flexible, modular and easy to evolve.

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 cloud-enabled elasticity and rapid provisioning of resources.

Architecting with microservices, however, is not an easy task as it requires to manage a distributed architecture and its challenges, which include network latency and unreliability, fault tolerance, complex services’ orchestration, data consistency and transaction management, and load balancing. Cloud infrastructures and new architectures play a fundamental role for realizing microservice architectures and managing the associated challenges and complexities.

An illustrative example of a microservice-based architecture is Netflix.1 Netflix is implemented as an ecosystem of small, independently deployable, and independently scalable microservices. When requests come from client-side devices (e.g., smartphones, TVs, laptops), they reach the Netflix API orchestration service, which acts as an API gateway towards the rest of the ecosystem (i.e., it exposes a set of coarse-grained APIs and then routes incoming requests to the specific target microservices within the ecosystem). The Netflix API implements the logic to route, sequence, and parallelize incoming calls from the devices. Microservice-based systems must be designed to cope with failure (Fowler and Lewis, 2014), meaning that the application must be able to tolerate possible service failures either by recovering as fast as possible or by gracefully degrading its functionalities. At Netflix, this is achieved by having each microservice manage its own layer of persistence (as independently as possible from the other microservices), by pushing as much as possible towards stateless microservices, relying on real-time monitoring, using libraries for fast recovery of services, and by implementing the so-called circuit breaker architectural pattern for avoiding cascading failures (e.g., via the open-source Hystrix2 library). Further details about the Netflix microservice-based architecture can be found in the Netflix technical blog3.

3. Study design

In this research we follow the well-established guidelines for systematic mapping studies (Petersen et al., 2015; Kitchenham and Brereton, 2013). In this section we present the key aspects of the design of our study.

3.1. Goal and research questions

The goal of this study is to identify, classify, and evaluate the trends, focus, and potential for industrial adoption of existing research in architecting with microservices from a researcher’s and practitioner’s point of view. This abstract goal can be refined into the following research questions.

RQ1: What are the publication trends of research studies about architecting with microservices?

Rationale: academic research is a dynamic ecosystem, where a multitude of researchers and research groups investigate on specific scientific problems over time with different degrees of independence and different methodologies. Relevance for researchers: the results of this research question help researchers in (i) quantifying the intensity of scientific interest on architecting with microservices, (ii) identifying the academic venues where related papers about architecting microservices are published, and (iii) identifying the academic venues where new results about architecting microservices may be better received (and appreciated) by the scientific community.

Relevance for practitioners: the results of this research question help practitioners in identifying the relevant venues where scientific knowledge is created, so to (i) take inspiration for solving problems which have been already targeted by researchers, (ii) get a more orthogonal and cross-organizational perspective with respect to architecting with microservices, and (iii) identify the research groups which are prominently contributing in the field, so to catch future collaboration opportunities.

RQ2: What is the focus of research on architecting with microservices?

Rationale: architecting microservices is a multi-faceted research topic, where researchers can focus on very different aspects (e.g., continuous integration, performance analysis, security, deployment elasticity, monitoring and fault tolerance), applying very different research methodologies (industrial case studies, empirical evaluations, feasibility studies, etc.), and providing different types of contributions (e.g., specific architectural tactics, architectural languages, etc.). Relevance for researchers: by answering this research question we support researchers by providing (i) a solid foun-

1 https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/
2 https://github.com/Netflix/Hystrix
3 http://techblog.netflix.com,
3.2. Search and selection process

In the following we present the stages of our search and selection process (see Fig. 1).

1. **Initial search.** In this stage we performed\(^4\) automatic searches on electronic databases and indexing systems. The selection of these electronic databases and indexing systems was

\[ \text{(architect}^* \text{ OR design}^* \text{ OR system OR structure}^*) \text{ AND} \]
\[ \text{(microserv}^* \text{ OR micro-serv}^* \text{ OR "micro serv"}^*) \]

Listing 1. Search string used for automatic research studies.

guided by: (i) the fact that they are the largest and most complete scientific databases and indexing systems in software engineering (Petersen et al., 2015; Kitchenham and Brereton, 2013), (ii) the fact that they have been recognised as being an effective means to conduct systematic literature studies in software engineering (Petersen et al., 2015), (iii) their high accessibility, and (iv) their ability to export search results to well-defined, computationally amenable formats. Our search string is shown in the listing below.

For consistency, the search string has been applied to title, abstract and keywords of papers in all electronic databases and indexing systems 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, book series, etc. In this stage we manually removed such impuris results from our set.

3. **Merging and duplicates removal.** In this stage all relevant results from the first stage have been combined together into a single dataset.

4. **Application of selection criteria.** We considered all the selected studies and filtered them according to the following inclusion and exclusion criteria.

I1 - Studies focussing on architectural solutions, methods or techniques (e.g., tactics, patterns, styles, views, models, 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, controlled experiment, exploitation in industry, simple examples, etc.).

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 focusing only on low-level technological aspects, the inner details of microservices, etc.).

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.

Even if secondary studies have been excluded because of the E3 exclusion criterion, we considered them in our study for: (i) checking the completeness of our set of primary studies (i.e., if any relevant paper was missing from this study); (ii) identifying important issues to be analysed in this study; (iii) defining what is the contribution of this study to the literature (see Section 9).

5. Snowballing. In this phase we complemented the automatic search with a closed recursive backward and forward snowballing activity (Wohlin, 2014). In the backward snowballing we focussed on all the references of each considered study, whereas for the forward snowballing Google Scholar has been used to obtain those studies citing the current one (Wohlin, 2014).

6. 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. For example, if a primary study was published in more than one paper (e.g., a conference paper extended to a journal version), only one instance has been considered as a primary study. Generally, the journal version has been preferred, since more complete, but both versions have been used in the data extraction and analysis of the publication trends (RQ1) phases. This step is necessary for ensuring completeness and traceability of the results (Wohlin et al., 2012).

3.3. Data extraction

In order to have a rigorous data extraction process and to ease the management of the extracted data, a well-structured classification framework has been rigorously designed. The resulting classification framework is shown in Fig. 2 and it is composed of three facets, each of them addressing its corresponding research question.
In the following we describe each facet of our classification framework.

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 of our classification framework. Goal of the keywording process is to develop a classification framework so that it fits the primary studies and takes their research focus into account (Petersen et al., 2008). The following details each step of the process depicted in Fig. 3:

1. Identify starting set of studies. Two researchers (R1 and R2) randomly extracted 5 studies, which have been used as pilot studies.

2. Identify keywords and concepts. Three researchers (R1, R2, and R3) collected keywords and concepts by reading the full-text of each starting study.

3. Cluster keywords and form categories. Two researchers (R1 and R2) clustered the collected keywords and concepts into a set of emerging categories. The output of this stage is the initial version of the classification framework. Examples of emerging categories include: supported architecting activities, scope in the software lifecycle, considered design patterns, considered quality attributes (e.g., performance, reliability, security), etc. Next steps have been performed for each primary study.

4. Extract data from current study. A researcher (R1) extracted information about the current primary study to be analysed and (i) collected information according to the parameters of the classification framework and (ii) collected any kind of additional information that was considered relevant and that did not fit within any parameter of the classification framework. If the collected information about the current primary study fit completely within the classification framework, then we proceeded to analyze the next primary study; otherwise, the classification framework was refined (this step involved three researchers, R1, R2, and R3).

5. Refine comparison framework. Two researchers (R2 and R3) 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 in order to make it a better fit with the primary studies.

The above described process ended when no primary study to analyze was left. The specific parameters emerging from the keywording process are independent from each other and have been extracted independently; they are described in Section 5. Finally, in this phase we agreed that 9 analysed studies were semantically out of the scope of this research, so they have been excluded.

Potential for industrial adoption (RQ3). In order to analyse the potential for industrial adoption of microservices, we have classified and extracted five 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, (iv) open-source test system for identifying existing benchmarks for microservice architectures, and (v) number of microservices used for evaluation.

3.4. Data synthesis

Our data synthesis activity can be divided into three main phases: vertical analysis, trend analysis, and horizontal analysis. When performing vertical analysis, we analyzed the extracted data to find trends and collect information about each parameter of our classification framework. When performing Trend analysis, we focused on how each possible value of all parameters of the classification framework evolves over time. When performing horizontal analysis, we used contingency tables for evaluating the actual existence of relations across different parameters of the classification framework, we made comparisons between pairs of parameters, and we identified perspectives of interest. In those phases, 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).

3.5. Replicability of the study

To allow easy replication and verification of our study, a complete replication package Di Francesco et al. (2017a) is publicly available to interested researchers. Our replication package includes: the detailed research protocol of this study, a document
providing a precise definition 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 checking, analyzing, and visualizing the extracted data.

4. Results - Publication trends (RQ1)

In this section we present the results we obtained when analyzing the publication trends on architecting with microservices. In order to provide a complete picture about the number and types of publications on the topic, in this section we consider all the selected publications, independently of the combination step we performed during the search and selection process (see Section 3.2). More specifically, for answering RQ1 we consider the total set of 119 primary studies, which includes both the entire set of 103 primary studies and the 16 primary studies resulting from the combination activity.

4.1. Obtained results (RQ1)

Publication years. Fig. 4 presents the distribution of publications on architecting with microservices over the years. The year 2017 is highlighted with a grey background to remark that data within this period is only partial, as the search and selection process was performed in May 2017.

The Figure emphasizes a clear confirmation of the scientific interest on architecting with microservices in the years 2015 through 2017. A very small number of publications have been produced until 2014, which is actually the first year in which (i) microservices started to attract the interest of large organizations, and (ii) the term microservice as architectural style was consistently used (Pahl and Jamshidi, 2016). As a confirmation, even if the six studies published before 2014 were about systems composed of small-scale lightweight services (P9, P60, P61, P62, P104, P105), they were referring to slightly different perspectives on microservices as they are considered today. For example, P9 considers microservices as low-level software components in the robotic domain, whereas P60 considers microservices as mobile services generated by end-users. Year 2015 signed a booming in the research field of architecting with microservices, with the trend increasing in 2016 and still growing in the first months of the year 2017.5

Publication types. Fig. 4 shows the publication types of the primary studies over the years. The high number of conference and journal papers indicate that architecting with microservices is progressing as research topic despite its relative young age; the relatively low number of workshop papers indicate that researchers commonly target more scientifically-rewarding publication types (like journals and conferences) when working on architecting with microservices.

Publication venues. We can observe an extreme fragmentation in terms of publication venues, where research on architecting with microservices is spread across 91 venues spanning different research areas like cloud infrastructures, software engineering, software services, autonomic computing, software maintenance, etc. This result indicates that architecting with microservices is considered as an orthogonal research target with many cross-cutting concerns. In Table 1 the most targeted publication venues are reported. We can notice that researchers are mainly targeting specialized venues on architecting with microservices (i.e., AMS), cloud computing venues (i.e., IEEECC) and software architecture venues (i.e., ICSEA). Researchers and practitioners can consider those venues as their starting points for their exploration into the state of the art on architecting with microservices.

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. (2006). We chose this comparison because (i) it has been widely used in various systematic mapping studies (e.g., Engström and Runeson, 2011; Mehmood and Jawawi, 2013; Petersen, 2011), and (ii) its categories are quite cost-effective to be identified by reading a paper without going into its very details (Petersen et al., 2008).

As shown in Fig. 5, here the clear winner is solution proposal (86/119). This result is due to the fact that the microservice architectural style is still in its infancy (we recall here that its 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 5.1 for the details on which problems are targeted). Validation research (43/119) is the second most recurrent research strategy, highlighting the fact that researchers are actually providing some level of evidence about their proposed solutions, e.g., by simulations, in-

5 Our search process covers the research studies published until 8th May 2017.
the-lab experiments, prototypes, etc. At the other end of the spectrum, evaluation research is performed very rarely (1/119), meaning that industry- and practitioners-oriented studies (e.g., industrial case studies, action research, practitioner-targeted surveys) are not yet in the main focus of researchers today. Specifically, in P34 a case study conducted in a software company has been presented; in this context, the authors developed a Java application using both the monolithic approach and the microservice pattern. The fact that evaluation research is rarely performed has a negative impact on the potential for transferring current research results in industry. This suggests a gap that should be filled by future research on architecting with microservices, especially if we want to either (i) solve real problems coming from industrial scenarios, or (ii) push further the technology transfer of research results in industry.

4.2. Trend analysis (RQ1)

In this section we report our analysis of the research trends over the years for the parameters related to RQ1. We have analyzed the research trends for all the parameters of the classification framework. However, some trends have been already discussed during the vertical analysis (e.g., the spike of conference publications in 2015 and 2016, see Fig. 6(a)) or do not have enough data points (e.g., the use of architectural languages – see Section 5.2), so in the following we focus exclusively on the most essential aspects we could observe. For the sake of completeness, we include in the replication package (Di Francesco et al., 2017a) of our study the figures showing the research trends for all parameters of the classification framework. In each figure, we have highlighted with a grey background the year 2017 to recall that the data within this period is partial, as the search and selection process was performed in May 2017.

For what concerns research strategies, we observe a growth of solution proposals from 2014. Studies proposing validation research are following the trend of solution proposals, but with a lower magnitude. Unfortunately, as previously discussed, evaluation research (i.e., the one involving industry- and practitioners-oriented research methodologies) is still lagging behind and its

<table>
<thead>
<tr>
<th>Venue</th>
<th>#Studies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workshop on Architecting with MicroServices (AMS)</td>
<td>5</td>
</tr>
<tr>
<td>IEEE Cloud Computing (IEEECC)</td>
<td>5</td>
</tr>
<tr>
<td>International Conference on Software Architecture (ICSA)</td>
<td>4</td>
</tr>
<tr>
<td>Service-Oriented System Engineering (SOSE)</td>
<td>3</td>
</tr>
<tr>
<td>Information Integration and Web-based Applications and Services (iiWAS)</td>
<td>3</td>
</tr>
<tr>
<td>Innovations in Clouds, Internet and Networks (ICIN)</td>
<td>3</td>
</tr>
<tr>
<td>International Conference on Performance Engineering (ICPE)</td>
<td>2</td>
</tr>
<tr>
<td>XSEDE Conference on Diversity, Big Data, and Science at Scale (XSEDE)</td>
<td>2</td>
</tr>
<tr>
<td>International Conference on Cloud Engineering (IC2E)</td>
<td>2</td>
</tr>
<tr>
<td>Enterprise Distributed Object Computing Workshop (EDOCW)</td>
<td>2</td>
</tr>
<tr>
<td>Cloud Computing Technology and Science (CloudCom)</td>
<td>2</td>
</tr>
<tr>
<td>EAI Summit on Smart Cities (RAI)</td>
<td>2</td>
</tr>
<tr>
<td>European Conference on Service-Oriented and Cloud Computing (ESOCC)</td>
<td>2</td>
</tr>
<tr>
<td>International Conference on Web Engineering (ICWE)</td>
<td>2</td>
</tr>
<tr>
<td>Utility and Cloud Computing (UCC)</td>
<td>2</td>
</tr>
<tr>
<td>Business Modeling and Software Design (BMSD)</td>
<td>2</td>
</tr>
<tr>
<td>CEUR</td>
<td>2</td>
</tr>
<tr>
<td>Other</td>
<td>74</td>
</tr>
</tbody>
</table>

Fig. 6. Trend analysis (RQ1).
trend over the years seems not to be very promising. This confirms the urgency to fill the gap with respect to the industrial relevance of the performed evaluations.

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.
- Only one study applied industry- and practitioners-oriented research methodologies (e.g., industrial case studies, action research), leaving a gap with respect to the industrial relevance of the performed evaluations.

5. Results - research focus (RQ2)

As described in Section 3.3, 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 with microservices, namely scope of the research (see Section 5.1) and support for architecting (see Section 5.2).

5.1. Scope of the research

With this category we provide information to help researchers and practitioners in putting into context research studies on architecting with microservices. In the following we discuss the obtained results.

Target problems. Fig. 7 presents the problems targeted by the primary studies. The obtained 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. Interestingly, the bottom area of Fig. 7 shows problems that are related to system-level aspects like time to market, low testability, low portability, and security. Moreover, only one paper (P64) is addressing the problem of benchmarking microservice-based applications. These aspects have been extensively investigated in the software architecture area, but are still new to microservice architectures; this result is an indicator of a potentially relevant research gap needing attention in the future.

Research contribution. By referring to Fig. 8, the realization of microservice-based application and the consistent number of method studies may indicate that the complexity in the realization of these systems is still very high. Interestingly, few papers are investigating architectural languages and design patterns for microservices, unveiling interesting gaps to be filled by the research community.

Main research area. The main research area is about the principal area of interest of the research to which the primary study belongs, e.g., cloud computing, system migration (see Fig. 9). The focus on the system quality (e.g., performance, maintainability) suggests that the microservice architectural style has direct impact on the design of a system and that researchers are still investigating how to leverage its characteristics.

A significant attention is also given to the use of microservices in cloud environments. 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. An industrial survey on the activities and the challenges of migrating towards microservices (Di Francesco et al., 2018) provides insights to this topic from an industrial perspective.

If on the one side microservice architecture have been applied to recent technologies like Internet of Things, mobile apps, and other domain-specific fields as robotics (P9) and datacenters (P28), on the other side a significant number of studies are focusing on other research areas (not reported in the figure), such as microservice architecture recovery (P85), distinguishing characteristics between microservice- and service-oriented architectures (P77, P84), deployment cost models definition (P79).

Abstraction layer. As shown in Fig. 13, microservices can run on top of (i) a physical machine running an operating system, (ii) a machine running a container engine, (iii) a machine running a virtualized environment (in this setting the hypervisor is mapped as operating system), or (iv) a machine running a container engine on top of a virtualized environment.

As shown in Fig. 10, more than half of the studies focus on the microservice layer only, without considering any other layer. This result is also aligned with the recent advent of serverless functions running in the cloud, where the developer is asked to provide the business logic that should run in the cloud, whereas the operational overhead is taken care by the platform (e.g., AWS Lambda).
In this context, serverless platforms are able to transparently manage infrastructural and operations aspects of the system, such as its deployment and configuration, monitoring and logging (at different levels, like operating system, containers, communication, etc.), security facilities and patches, operating system, platforms, and libraries updates, management of the services lifecycle, services vertical and horizontal scaling, and so on.

Differently, other studies not only focus on microservices, but also consider the environment as an important aspect of the architecture. More specifically, the container and the virtual machine layers were discussed respectively in 24 and 19 studies. This particular focus on containerization and virtualization confirms them as key enabling technologies for MSA. Moreover, a few studies (P16, P27, P39, P75) also consider the possibility to run containers on top of virtual machines, thus combining the resource utilization benefits of virtual machines and the portability and efficiency of containerization (Jaramillo et al., 2016), which seems to be particularly suitable for offering microservices on the IaaS cloud model (Khazaei et al., 2016).

**Software lifecycle scope.** As shown in see Fig. 11, the number of studies on design is significantly higher than the number of those focusing on other lifecycle phases. In 32 primary studies the microservice architectural style is related with operations of deployment and configuration of the environment needed for the services in general. We have also investigated which studies relate the microservice architectural style with DevOps. Among the 103 primary studies, 30 of them discussed DevOps with two slightly different perspectives. On the one side, some consider DevOps as the set of practices intended to reduce the time between committing a code changes and rolling it out in production, while ensuring high quality (Bass et al., 2015). On the other side, DevOps deals with the goal of having the development and operations teams work closely together for achieving rapid and continuous release cycles (Fazio et al., 2016). A total of 32 studies explicitly discussed on the requirements of the microservice approach/appli- cation included in the study. This helps define the specific con- text information the microservice architectures are subject to. As an example, in P103 the authors discuss both functional and non-functional requirements when they use a microservice architecture to address key practical challenges in smart city platforms. Finally, given the trend in the scope of the studies, we conjecture that the areas of microservices maintenance and testing will attract further research when the fields of design, implementation and operation will gain more maturity.

**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. As shown in Fig. 12, the most recurring definition was the one provided by Fowler and Lewis (2014), followed by the ones given by Newman (2015), and others. In 27 of the 103 studies, the authors have either provided their own definition of microservices or have used an informal definition. Nevertheless, the definitions provided by Lewis & Fowler and Newman seem to start prevailing over other definitions.

5.2. Support for architecting

We characterize primary studies with respect to how they support architecture-specific concerns and activities, such as design patterns, support for specific quality attributes, recurrent infrastructural services.

**Architecting activities.** We have based our classification of architecting activities according to the introvert/extrovert nature of software architects discovered in Malavolta et al. (2013). The introvert nature regards the analysis and design of the software activities. It has been refined into the architecting activities defined by Li et al. (2013). The extrovert nature regards the communication between architects and other stakeholders. It has been further classified into the providing information and getting input parameters proposed by Kruchten (2008). Highlighted in darker gray in the figure, we can also observe how little investigation has been performed on extrovert architecting activities, i.e., providing information and getting input from other stakeholders of the system. From a research perspective, the low interest in these complementary activities indicates that there are areas of improvement in the engagement of customers and users, and also in the project management and communication with teams.

**Quality attributes.** Fig. 15 shows that performance, maintainability, and functional suitability are by far the most investigated quality attributes, while the remaining qualities are almost equally represented. Among the primary studies discussing performance (59/103), we have classified which of them have a special focus on scalability. It resulted that scalability aspects are addressed in 36 out of 59 studies, suggesting that many researchers seem to consider scalability as a sub-problem of performance when architecting with microservices.

**Architecture provenance.** According to our classification framework, an architecture is designed if it is created prior its implementation, otherwise it is considered as an extracted architecture. In Fig. 16, the overwhelming focus on design suggests that it is not easy to realize microservice architectures unless an actual analysis and design of the system is performed prior to its implementation.

**Architectural language.** An architectural language can be considered as any form of expression used for architecture de-
Cloud Applications (TOSCA) (2013) standard has not been used by any of the primary studies for designing microservice architectures. TOSCA is a standard that can be used for representing portable cloud applications and supporting their life-cycle management (Bergmayr et al., 2018), and is a promising candidate for the microservice architectures (Ruiu et al., 2016; Lipton et al., 2018; Shalom, 2017).

From a researcher’s point of view, the use of informal architectural languages and the lack of a predominant architectural language may lead to difficulties in the description and modeling of microservice architectures. We can conjecture that this concern can be addressed by working on a standard architecture language, which may help in having a shared common, industry-proven representation for microservice architectures. Proposing an architectural language for microservices helps architects in many activities; for example, it can help in reasoning about the system as a whole, performing analyses on the system qualities, coping with the dynamic and changing aspects of the application at runtime. Furthermore, an architectural language is a powerful communication instrument to enable better communication with both technical (e.g., developers, architects) and non-technical stakeholders (e.g., customers and users) at the right level of abstraction and with a shared technical vocabulary.

**Architecture description types.** Fig. 17 shows that the architectures proposed were mostly described in terms of their structural aspects, while the behavioral aspects were addressed less often. The major focus of researchers on static rather than dynamic aspects gives another perspective about certain types of challenges inherently related to the definition of the microservices, as for example finding the proper level of granularity of each service or migrating legacy systems.

**Technology-specific.** We have classified as technology-specific the studies proposing solutions, methods or techniques that are specific to one or more particular technologies (e.g., Docker). A total of 75 primary studies were not technology-specific, while the remaining 28 studies were technology-specific. The predominance of not technology-specific studies is a good indicator because approaches and solutions can be reused across technologies. Differently, technology-specific studies bear the advantage of being more detailed, but their applicability and portability in the future might be limited. In the set of technology-specific studies, Docker is clearly
the most recurring technology (12/28) while other technologies are quite scattered, with a few occurrences of Java EE (P72, P78), Spring framework (P19, P63), Eureka (P19, P28), and other specific technologies (e.g., Serfinode (P31), KVM (P23)).

**Design patterns.** Each design pattern has been identified as reported in the primary studies, thus each pattern has to be considered disjoint from the others (e.g., if the API Gateway is used for implementing the load balancing pattern, we report each as a separate pattern). The set of discussed patterns is reported in Fig. 18. The most recurring design patterns when architecting with microservices are: API gateway, Publish/subscribe, Proxy, Circuit breaker, and Discovery patterns.

It is important to note that 7 primary studies (P10, P18, P29, P36, P42, P44, P48) have addressed or referred to a set of design patterns which have not been discussed in the other studies. In P10, four different patterns for implementing loose coupling in microservices are reported; namely: location independence, communication independence, security independence, and instance independence patterns. In P29, the authors report about the ports and adapter pattern, also known as hexagonal architecture ([Cockburn, 2007]), and the immutable server pattern ([Morris, 2014]). In P36 a set of data adapter patterns for working with data provision mechanisms are addressed. In P42, the bulkhead pattern is presented to support fault isolation within a microservice. In P44, cloud-focused patterns such as the Twelve-Factor App ([Wiggins, 2014]), and cloud computing patterns are referred. In P48, authors not only address several existing patterns, but they propose a new pattern called the database-is-the-service.

**Infrastructure services.** These are the infrastructure services supporting non-functional tasks, as defined by [Richards (2015)]. As shown in Fig. 19, microservice architectures, being inherently distributed, show a clear need for monitoring capabilities (e.g., logging, profiling) but also for system level management (e.g., health management, autoscaling) in order to leverage the underlying infrastructure efficiently. A significant research interest is pointing to service brokering and service orchestration, which confirms that service management capabilities are fundamental to this area.

5.3. Trend analysis (RQ2)

Fig. 20 summarizes our results for the trend analysis related to RQ2. In the following we will discuss only the most relevant trends.

When looking at the main research areas (see Fig. 20(c)), since 2015 we are seeing a growth of system-level quality, microservices in the cloud, and migration; we can conjecture that this trend will continue in the next years. Microservices in the context of mobile-enabled systems has a negative trend, meaning that in the last years researchers on architecting microservices seem to be less interested in microservices deployable on mobile devices in favour of microservices deployed in the back-end of mobile-based systems. This trend is in line with the classical definition of microservice, which is heavily influenced by concepts coming from containerization and cloud computing.

Research on microservices has a clear trend when considering the software lifecycle scope (see Fig. 20(e)), researchers are increasingly focussing on the design of microservice-based systems, followed by operations and implementation. Interestingly, requirements is recently starting to attract researchers’ attention, hence suggesting that this trend might continue in the next years.

Looking at Fig. 20(g) (architectural activities), it is interesting to observe a spike in the focus on architectural analysis in 2015 and 2016, unveiling the fact that researchers are devising and applying (new) architecture analysis techniques in the last years. We believe that reasoning at the architectural level of abstraction allows those techniques to be applicable on large scale systems like the microservice-based ones.

It is clear that performance is the raising star in terms of quality attribute (see Fig. 20(h)), followed by a relatively strong interest over the years on maintainability and functional suitability. From the collected data, we can also observe that the scientific interest in security and usability is decreasing after 2015. We conjecture that the high degree of isolation provided by containers (e.g., by using Dockers namespaces) and resources limitations enforced in virtualized environments (e.g., limits for CPU load, I/O access, memory usage, and networking) may have played a role in this context.

When looking at infrastructure services (Fig. 20(l)) we notice that monitoring (e.g., logging, profiling) and system level management (e.g., autoscaling, load balancing) have the most prominent growth in the last years. These are clearly the two types of infrastructure services that are attracting the strongest attention of researchers. We expect that this trend will continue in the future.

---

**Main findings:**

- Research scope involves problems that consolidate the need to master the tradeoffs between complexity and flexibility; here we can notice a strong focus on cloud and mobile paradigms, and legacy migration. Benchmarking is growing in 2017, potentially unveiling a promising future research direction. Requirements are starting to attract researchers’ attention in recent years.
- Architecture analysis emerges as the most popular architectural activity. Results suggest software architecture as a powerful instrument for stakeholder engagement. Extrovert activities are raising since 2015, even if they are still not mainstream.
- The clear focus on infrastructure services has the potential to help devising new related patterns and styles and hence further leveraging cloud-based architecture models. In the set of investigated infrastructure services we observed that monitoring and system level manage-
Fig. 20. Trend analysis (RQ2).
ment (e.g., health management, autoscaling, load balancing) have the most prominent growth in the last years.

An industrial standard describing the architecture of microservice-based systems does not yet exist. If present, it could help support the architecting activities of microservice-based systems better. The most promising standard in this direction is the OASIS standard for Topology and Orchestration Specification for Cloud Applications (TOSCA) that, with the proper customization, could be used in the future to model microservice architectures (Lipton et al., 2018).

6. Results - potential for industrial adoption (RQ3)

6.1. Obtained results (RQ3)

Readiness level. Defined by the systematic measurement system for assessing the maturity of a particular technology (Mankins, 1995), the technology readiness level (TRL) is an integer \( n \) where \( 1 = n = 9 \). This measure has been used by the Horizon 2020 European Commission for the 2014/2015 work program.\(^7\)

We have classified the TRL of each primary study to emphasize the environment in which the proposed approach has been validated. Specifically, in the context of this study we classify the TRL of each primary study on a 3-level scale: (i) low TRL (i.e., TRL \( \leq 4 \)) means that a technology is either formulated, validated or demonstrated at most in lab-based environments, (ii) medium TRL (i.e., \( 5 \leq \text{TRL} \leq 6 \)) means that a technology is either validated or demonstrated in industrially relevant environment, and (iii) high TRL (i.e., TRL \( \geq 7 \)) means that a technology is either completed, demonstrated, or proven in operational environment. Fig. 21 presents the TRL levels of our primary studies.

The obtained results indicate that (i) research on architecting with microservices is still in its initial phase for what concerns the transferability of the developed technologies to industry and (ii) there is a relatively large number of studies (9/103) (P3, P7, P19, P33, P35, P59, P68, P83, P87) in which the actual system has been proven in its operational environment (TRL = 9).

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 Fig. 22, the results are encouraging, as in almost half of the primary studies (42/103), there is the involvement of at least one industrial researcher or practitioner; this suggests some knowledge exchange between academia and industry.

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 (Jaccheri et al., 1998). Based on the given definition, we categorize a tool either as software-based or knowledge-based. Overall, 54 primary studies provided software-based tools and 77 primary studies provided knowledge-based tools.

From a research point of view, this result indicates the need to support knowledge-based tools with more software-based tools in order to demonstrate how effective knowledge-based tools are and how they can be compared one another. This can help researchers and practitioners to improve the overall quality of microservice-based systems.

It is important to notice that this parameter can be related to the Research contribution parameter of our classification framework (see Section 5.1). Indeed, here we are focussing on whether the proposed approach is proposing a specific tool, procedure, or guideline (or a combination thereof), as opposed to the main research contribution, which may be about the description of a problem, a new application of the microservices architectural style, etc.

**Open-source test system.** When screening the 103 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 system called Acme Air, which was used and discussed in two different primary studies (P26, P85). 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 researchers and practitioners with a very useful benchmark for evaluating, measuring, and comparing their own solutions over a common reference system. Acme Air is publicly available as open-source repository on GitHub.8

The lack of practical systems for benchmarking microservice-based architectures can severely impact the knowledge transfer from academia to industry. A first step in this direction has been performed with the Acme Air system, which however is still far away from being a realistic benchmarking system. Specifically, it is composed of only six services and all of them are developed using the same programming language and underlying platform; this is rarely the case in real microservice-based systems, for example the Netflix software stack is composed of more than 20 technologies,9 such as Python, Node.js, Java, React, MySQL, PostgreSQL, Cassandra, and Hadoop. In the near future it will be fundamental for researchers to have a shared, technologically polyglot, open-source benchmarking system that can be used for testing their proposed solutions and for increasing the readiness level of their research products. Technically it is also possible to (semi-)automatically generate large scale systems composed of a large number of heterogeneous microservices; even if a generated system may be realistic only from a syntactical and scale perspective (i.e., its microservices communicate with each other, but they do not do any meaningful operation from a semantic point of view), it may already prove useful for researchers focussing on dependability aspects like scalability, performance, security, availability.

**Number of microservices used for evaluation.** Most of the primary studies have only used a relatively small number of microservices for their evaluations (i.e., less than 10). Only three primary studies (P82, P35, P72) have used a relatively significant number of microservices using a total of 27, 28 and 67 microservices respectively. In order to put this result into context, a recent industrial survey (Di Francesco et al., 2018) showed that the expected number of microservices deployed after migrating towards the microservices architectural style varies between 5 and 250, with an average of 59. In the future, if the research community on microservice architectures aims to bring new emerging approaches to maturity and perform realistic evaluations, the number of microservices used for evaluation purposes should be increased.

6.2. Trend analysis (RQ2)

Fig. 23 summarizes our results for the trend analysis related to RQ2. In particular, we notice interesting trends with respect to industry involvement (Fig. 23(b)). Firstly, academic-only publications are increasing at a fast pace since 2014 and publications with both academic and industrial researchers are growing in the last two years as well. Finally, publications with only industrial authors are decreasing, potentially in favour of publications where also academic researchers are involved.

**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 studies with high TRL are quite heterogeneous, and their contributions range from a component-based gateway middleware (P2), to auto scaling services (P33), to the management of mobile and IoT workloads (P52), etc. All studies with high TRL involve an industrial case study or an application of the proposed solution into on an industrial-scale system.
- The balanced involvement of industrial and academic authors, however, is promising for knowledge co-creation and cross-fertilization.
- The industrial relevance of research evaluations shall be fostered by having more significant open-source test systems or benchmarking applications available.

7. Orthogonal results

Table 2 presents the results of our horizontal analysis. In this phase of the study, we firstly automatically computed a contingency table for every possible pair of parameters of our classification framework. Then, we collaboratively created and discussed a set of 37 potentially relevant insights to be investigated. We iteratively analyzed each potentially relevant insight created in the previous step in order to check if its contingency table actually confirms or disproves its related hypotheses. Finally, we filtered out all the results which were either (i) not supported by a sufficient number of data points, or (ii) chaotic, not revealing any evident pattern. This filtering step was performed manually and collaboratively by three researchers until reaching a full agreement. The full list of potentially interesting relations and the contingency tables for evaluating the actual existence of those relations are available in our replication package (Di Francesco et al., 2017a).

**Main findings:**

- Gaps for future research especially point toward security and real-time communication in specific areas like IoT and mobile.
- With an eye on quality: (i) many orthogonal results suggest that quality control and security are attracting insufficient research. Given the substantial investments in modernizing software solutions with microservices, this can become a real issue in industrial practice. Also, (ii) the quality attributes performance and maintainability occur with striking frequency together with various target problems like low flexibility, low efficiency, complexity, and modernization.
- With the pervasive coverage of the design lifecycle phase, a few design patterns seem to consolidate towards a catalogue of solutions ready for reuse by practitioners.

---

8 https://github.com/acmeair/acmeair
9 http://stackshare.io/netflix
Indeed, the presence of reports in the systematic literature is mitigated to a great extent by a presence of systematicity in the selected research, which helps to validate the results of such studies. Hence, we can consider these approaches as a valid way to address the research problem of interest.

All the selected studies have a high validity score, which is determined by the number of papers and the proportion of each of them. In our case, we have selected studies from the systematic literature, which have a high validity score of 80%. This means that we can be confident in the validity of the results of our study.

However, it is important to note that the selection criteria for our study are subjective and might introduce a degree of selection bias. The criteria used in the selection of studies are as follows:

1. **Academic** - focused on academic papers
2. **Industrial** - focused on industrial papers
3. **Mixed** - focused on both academic and industrial papers

In conclusion, our study has selected studies from the systematic literature, which have a high validity score. This means that we can be confident in the validity of the results of our study.

8. Threats to validity

In 2015, Petersen et al. (2015) 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 versus 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 of our study and how we mitigated them (Figs. 3-23 and A.1).

External validity. The most severe potential external threat to the validity of our study is on our primary studies not being representative of the state of the art on designing microservices. To avoid this to happen, we applied a search strategy consisting of both automatic search and backward-forward snowballing on the selected studies in combination. Specifically, we mitigated the presence of potential gaps left out by the automatic search (which is intrinsically syntactic) by means of the snowballing technique. Indeed, as recommended in the most recent guidelines for systematic studies (Petersen et al., 2015), we extended the coverage of the automatic search by complementing it with a snowballing activity, thus enlarging the set of relevant studies by considering each study selected in the automatic search, and focussing on those papers either citing or cited by it. Also, we considered only peer-reviewed papers and excluded the so-called grey literature (e.g., white papers, editorials, etc.). This potential bias did not impact our study significantly since considered papers have undergone a rigorous peer-review 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. Specifically, we thoroughly discussed the definition of each selection criterion in order to have a minimal, but complete set of selection criteria, according to the goal of our study. It is important to note that we decided to have the E2 selection criterion while piloting the search string on the electronic data sources. In that phase, we noticed that a large number of research articles used a very simple example; in those cases, microservices are outside the focus of the proposed research (e.g., approaches for self-adaptive systems which can be applied to any type of system, approaches focussing on REST APIs in general, etc.) and the microservices domain has been used by the authors of the articles in order to contextualize their research contributions in a more recent technology. We added the E2 selection criterion to avoid this phenomenon. Nevertheless, we are aware that the E2 selection criterion could have been risky in case of abuses, so during the application of the selection criteria we have been extremely rigorous and, when in doubt, we went through the full text of the whole study being considered. Moreover, secondary studies have been excluded (criterion E3) because they are meta-studies, and they provide a different perspective about microservices w.r.t. primary studies, which focus more on proposing specific architectural solutions, methods, or...
### Table 2: Orthogonal results.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target problems - Main research areas</strong></td>
<td>When looking at the distribution of main research areas over target problems we notice that service composition, resources management, low flexibility, and complexity are well covered by all the main research areas. Differently, the least covered problems are: benchmarking (only 1 perspective over 7), low testability (2/7), security (3/7), real-time communication (3/7), low portability (3/7), and time to market (4/7). The research community in software architecture can consider those problems as potential good candidates for contributing in solving not-yet-explored challenges in architecting with microservices. Moreover, some interesting research gaps for the research community on microservice architecture emerged: security from the IoT perspective, real-time communication from the cloud, IoT and mobile perspectives, low auditability from the IoT and mobile perspectives, and data management from the cloud perspectives. Those problem-perspective pairs have never been investigated in any of the analyzed primary studies.</td>
</tr>
<tr>
<td><strong>Target problems - Readiness levels</strong></td>
<td>There are some problems in which the technology readiness is still leaning towards lower values. These are: runtime uncertainty, modernization, and low portability. Those problems can be considered as potentially relevant for future researchers as there seems to be a barrier to overcome for architecting with microservices with runtime uncertainty and high portability.</td>
</tr>
<tr>
<td><strong>Target problems - Software lifecycle scopes</strong></td>
<td>In general, design is the most frequently considered lifecycle phase independently of the considered target problem (the only exception is low testability with zero occurrences), followed by implementation (only exceptions are low testability and benchmarking with zero occurrences). During maintenance almost all target problems are considered, with no clear interesting trends. Requirements are mainly considered when dealing with low flexibility problems (11 occurrences), complexity (9), low efficiency (8), and modernization (6). There are target problems which have been considered from a very narrow set of lifecycle phases, like: low testability (1/6, testing), benchmarking (2/6, design and testing), and security (3/6, design, implementation, operations). In the future it will be interesting to see if those extremely scoped problems will expand towards a larger number of lifecycle phases, such as requirements, implementation, and maintenance.</td>
</tr>
<tr>
<td><strong>Target problems - Quality attributes</strong></td>
<td>The most recurrent pairs of target problems and quality attributes are: low flexibility with performance (16 occurrences) and maintainability (14), low efficiency with performance (15), complexity with performance (12) and maintainability (13), service composition with performance (12), modernization with maintainability (10). The identified pairs show the interdependencies between architectural problems and quality attributes that have been investigated most frequently by the community. Nevertheless, some interesting gaps caught our attention, revealing potentially fruitful research lines for future research on architecting with microservices: (i) portability has not been considered or has been addressed either low auditability or data management, (ii) compatibility, portability, reliability, and security have not been considered when addressing real-time communication, and (iii) compatibility, reliability, and usability have not been considered when dealing with security.</td>
</tr>
<tr>
<td><strong>Quality attributes - Design patterns</strong></td>
<td>Among the most frequently used design patterns we can see that the API gateway is benefitting all quality attributes with a spike in maintainability (12) and performance (9), while publish/subscribe is strongly related to the performance (7), maintainability (7) and compatibility (5) quality attributes. The circuit breaker pattern is related to reliability and portability (4), whereas proxy is related to performance (5). Differently, the discovery patterns are related to maintainability and performance (3). These results can be used by practitioners as a catalogue of prepackaged solutions for gaining better quality of a microservice architecture, as potentially they have been already validated by the research community, or even evaluated in an industrial setting.</td>
</tr>
<tr>
<td><strong>Architecting activities - Architecture description types</strong></td>
<td>If on one hand structural descriptions cover all the architecting activities, on the other behavioural descriptions are only slightly used for describing reuse (1 occurrence) of architectural assets (e.g., design elements, decisions, patterns), getting input and providing information (3). While it is reasonable to consider the activity of reuse more linked to structural concerns of the architecture, it is important to note that missing a behavioural viewpoint can be a strong limitation since it may prevent the architect from reasoning with other stakeholders on the functionalities delivered by the system.</td>
</tr>
<tr>
<td><strong>Architectural languages - Architecture provenance</strong></td>
<td>Architectural languages are predominantly used for the design of the architecture of the system. Moreover, informal architecture descriptions are the only notations used when dealing with extracted architectures. This means that architectural languages (like UML) seem to be only used for designing the system, but are neither used for understanding nor analysing the current state of the architecture of the system (i.e., architecture extraction) – in spite of the strong focus on architecture analysis.</td>
</tr>
<tr>
<td><strong>Industry involvement - Research contributions</strong></td>
<td>Industrial contributions are mainly present in studies contributing with (i) an application of architectural methods, principles or tools (17 academic, 7 industrial, 8 mixed) or (ii) a reference architecture (1 academic, 3 industrial, 3 mixed). Differently, academia is spread in many types of contributions, with the largest difference with respect to industrial contributions in method (23 academic, 2 industrial, 10 mixed) and problem framing (12 academic, 3 industrial, 4 mixed). This result indicates that the latter types of contributions (i.e., method and problem framing) are the ones in which industrial participation is missing the most.</td>
</tr>
<tr>
<td><strong>Industry involvement - Research strategies</strong></td>
<td>Even though the majority of solution proposals are authored by academic-only authors (44/71), an encouraging result is the fact that 19 (out of 71) primary studies have been authored by a mixed type of researchers (i.e., both academic and industrial) and 8 (out of 71) by industrial-only authors. With the exception of one paper (P70), validation research always involves academic authors (academic-only in 38 occurrences, and mixed in 10 occurrences), which is another encouraging trend since in principle academic researchers can support industrial ones in setting up well-designed, reliable experiments by following known methodological guidelines.</td>
</tr>
</tbody>
</table>

**Techniques.** Nevertheless, even if secondary studies have been excluded because of the E3 exclusion criterion, we considered them in our study for checking the completeness of our set of primary studies, for identifying important issues to be considered in our study, and for defining what is the contribution of our study to the literature.

**Internal validity.** We rigorously defined our research protocol, and we iteratively defined the classification framework by rigorously applying the keywording process. The synthesis of the collected data has been performed by applying well-assessed descriptive statistics. Also, 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 in Wohlin et al. (2012), 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 doing so, 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 (Petersen et al., 2015; Kitchenham and Brereton, 2013; Wohlin et al., 2012). 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 checked and replicated by other researchers.

9. Related work

A systematic mapping on microservices was performed by Pahl et al. on a set of 21 primary studies from 2014 to 2015 (Pahl and Jamshidi, 2016). It is a classification of the research directions in the field and highlights the relevant perspectives considered by researchers. Our study differs from Pahl and Jamshidi (2016) as follows: (i) we apply a more comprehensive search process by considering studies published in any year up to 2017, 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 with microservices.

Alshuqayran et al. (2016) 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. Their work and ours can be considered as complementary, both cutting the topic of architecting with microservices from different perspectives. The main difference between the two studies is that ours considers different research questions, thus leading to different results, findings, and implications.

Dragoni et al. (2016) performed an informal survey on microservices. 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 with microservices, rather than on providing a narrative viewpoint on their historical, current, and future traits.

Kratzke and Quint (2017) conducted a systematic mapping study on cloud-native applications. The main outcome of that study is a clear definition of cloud-native applications, which are defined as “distributed, elastic and horizontal scalable systems composed of (micro)services which isolate state in a minimum of stateful components. The applications and each of their self-contained deployment unit are designed according to cloud-focused design patterns and operated on a self-service elastic platform”. Even though the subjects of their study is different from ours (i.e., cloud-native apps vs architecting with microservices), the two studies share the overall goal (building a comprehensive body of knowledge about a topic) and some parameters of the classification framework (e.g., research strategy, quality attributes, publication trends, etc.). Besides the difference in the considered subject, we also investigate on the potential for industrial adoption.

Finally, Bergmeyr et al. (2018) conducted a systematic literature review about cloud modeling languages (e.g., TOSCA). The study is motivated by the fact that existing modeling lan-
## Appendix A. Primary studies

Table A.1 reports the full list of the 103 primary studies.

### Table A.1

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Authors</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>A Reference Architecture for Real-time Microservice API Consumption</td>
<td>Cristian Gadea and Mirece Trifan and Dan Ionescu and Bogdan Ionescu</td>
<td>2016</td>
</tr>
<tr>
<td>P2</td>
<td>Apache Airavata As a Laboratory: Architecture and Case Study for</td>
<td>Suresh Maru and Marlon Pierce and Sudhakar Padmighantham and</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>Component-Based Gateway Middleware</td>
<td>Chathuri Wimalasena</td>
<td></td>
</tr>
<tr>
<td>P3</td>
<td>Synapse: A Microservices Architecture for Heterogeneous-database Web</td>
<td>Nicolas Viennet and Matthias Lécuyer and Jonathan Bell and Roxana</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>Applications</td>
<td>Geambasu and Jason Nish</td>
<td></td>
</tr>
<tr>
<td>P4</td>
<td>Emergent Software Services</td>
<td>Nicolas Cardozo</td>
<td>2016</td>
</tr>
<tr>
<td>P5</td>
<td>Bifrost: Supporting Continuous Deployment with Automated Enactment of</td>
<td>Gerald Schermann and Dominik Schoni and Philipp Leitner and Harald</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>Multi-Phase Live Testing Strategies</td>
<td>C. Gall</td>
<td></td>
</tr>
<tr>
<td>P6</td>
<td>Sustaining Runtime Performance While Incrementally Modernizing</td>
<td>Holger Knoche</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>Transational Monolithic Software Towards Microservices</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P7</td>
<td>Case Study: Microservice Evolution and Software Lifecycle of the XSEDE</td>
<td>Walter Scarborough and Carrie Arnold and Maytal Dahan</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>Portal API</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P8</td>
<td>TopoLens: Building a CyberGIS Community Data Service for Enhancing</td>
<td>Hao Hu and Xingshen Hong and Jeff Terstriep and Yan Y. Liu and</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>the Usability of High-resolution National Topographic Datasets</td>
<td>Michael P. Finn and Johnathan Rush and Jeffrey Wendel and Shaowen Wang</td>
<td></td>
</tr>
<tr>
<td>P10</td>
<td>Practical Use of Microservices in Moving Workloads to the Cloud</td>
<td>D. S. Linticum</td>
<td>2016</td>
</tr>
<tr>
<td>P11</td>
<td>Container and Microservice Driven Design for Cloud Infrastructure DevOps</td>
<td>H. Kang; M. Le; S. Tao</td>
<td>2016</td>
</tr>
<tr>
<td>P12</td>
<td>Towards Integrating Microservices with Adaptable Enterprise Architecture</td>
<td>J. Bogner; A. Zimmermann</td>
<td>2016</td>
</tr>
<tr>
<td>P13</td>
<td>Security-as-a-Service for Microservices-Based Cloud Applications</td>
<td>Y. Sun; S. Nanda; T. Jaeger</td>
<td>2015</td>
</tr>
<tr>
<td>P14</td>
<td>TeNOR: Steps towards an orchestration platform for multi-PoP NFV</td>
<td>J. F. Riera; J. Batallo; J. Bonnet; M. Dias; M. McGrath; G. Petralia;</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>deployment</td>
<td>F. Liberati; A. Giuseppe; A. Pietrabissa; A. Cenelli; A. Petrin; M.</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Trubian; P. Papadimitrou; D. Dietrich; A. Ramos; J. Melian; G. Xilouri;</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>A. Kourtis; T. Kourtis; E. K. Markakis</td>
<td></td>
</tr>
<tr>
<td>P15</td>
<td>Vendor Malware: Detection Limits and Mitigation</td>
<td>O. Iyse: K. J. Hole; C. Ottestad; O. Ytrehus; R. Aarseth; J. Tellnes</td>
<td>2016</td>
</tr>
<tr>
<td>P16</td>
<td>Leveraging microservices architecture by using Docker technology</td>
<td>D. Jaramillo; D. V. Nguyen; R. Smart</td>
<td>2016</td>
</tr>
<tr>
<td>P17</td>
<td>Scalable microservice based architecture for enabling DMTF profiles</td>
<td>D. Malavalli; S. Sathappan</td>
<td>2015</td>
</tr>
<tr>
<td>P18</td>
<td>The Design and Architecture of Microservices</td>
<td>A. Sill</td>
<td>2016</td>
</tr>
<tr>
<td>P19</td>
<td>JMesh – A Scalable Web-Based Platform for Visualization and Mining of</td>
<td>X. Mouy; P. A. Mouy; D. Hannay; T. Dakin</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>Passive Acoustic Data</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P20</td>
<td>CYCLOPS: A micro service based approach for dynamic rating, charging &amp;</td>
<td>S. Patanjali; B. Truninger; P. Harsh; T. M. Bohnert</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>billing for cloud</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P21</td>
<td>A Reusable Automated Acceptance Testing Architecture for Microservices</td>
<td>M. Rahman; J. Gao</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>in Behavior-Driven Development</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P22</td>
<td>Architecture of an interoperable IoT platform based on microservices</td>
<td>T. Vreisk; I. Cavrik</td>
<td>2016</td>
</tr>
<tr>
<td>P23</td>
<td>Performance Evaluation of Microservices Architectures Using Containers</td>
<td>M. Amaral; J. Polo; D. Carrera; I. Mohamed; M. Unuwar; M. Steinder</td>
<td>2015</td>
</tr>
<tr>
<td>P24</td>
<td>A microservices architecture for collaborative document editing enhanced</td>
<td>C. Gadea; M. Trifan; D. Ionescu; M. Cordea; B. Ionescu</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>with face recognition</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Microservices Architectures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P26</td>
<td>Workload characterization for microservices</td>
<td>T. Ueda; T. Nakaite; M. Ohara</td>
<td>2016</td>
</tr>
<tr>
<td>P27</td>
<td>Challenges in Delivering Software in the Cloud as Microservices</td>
<td>C. Esposito; A. Castiglione; K. K. R. Choo</td>
<td>2016</td>
</tr>
<tr>
<td>P28</td>
<td>Microservice-based architecture for the NRDC</td>
<td>V. D. Le; M. M. Neff; R. V. Stewart; R. Kelley; E. Fritzinger; S. M.</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Dascalogu; F. C. Harris</td>
<td></td>
</tr>
<tr>
<td>P29</td>
<td>Microservices approach for the internet of things</td>
<td>B. Bartini; F. Golatowski; D. Timmermann</td>
<td>2016</td>
</tr>
<tr>
<td>P30</td>
<td>Towards microservices architecture to transcode videos in the large at</td>
<td>O. Barais; J. Bourcier; Y. D. Bloemberg; C. Dion</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>low costs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P31</td>
<td>Distributed Systems of Microservices Using Docker and Serfnode</td>
<td>J. Stubbs; W. Moreira; R. Dooley</td>
<td>2015</td>
</tr>
<tr>
<td>P32</td>
<td>Microservice Based Tool Support for Business Process Modelling</td>
<td>S. Alpers; C. Becker; A. Oberweis; T. Schuster</td>
<td>2015</td>
</tr>
<tr>
<td>P33</td>
<td>Polygot Application Auto Scaling Service for Platform as a Service Cloud</td>
<td>S. R. Seelam; P. Dettori; P. Westerink; B. B. Yang</td>
<td>2015</td>
</tr>
<tr>
<td>P34</td>
<td>Evaluating the monolithic and the microservice architecture pattern to</td>
<td>M. Villamizar; O. Carcés; H. Castro; M. Verano; L. Salamanca; R.</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>deploy web applications in the cloud</td>
<td>Casallas; S. Gil</td>
<td></td>
</tr>
<tr>
<td>P35</td>
<td>Swiss TSO integrated operational planning, optimization and ancillary</td>
<td>D. Tchobourev; D. Wiczynski</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>services system</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P36</td>
<td>Experience on a Microservice-Based Reference Architecture for</td>
<td>M. Vanden; H. Lichter; A. Steffens</td>
<td>2014</td>
</tr>
<tr>
<td></td>
<td>Measurement Systems</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P37</td>
<td>Microservices and Their Design Trade-Offs: A Self-Adaptive Roadmap</td>
<td>S. Hassan; R. Bahsoon</td>
<td>2016</td>
</tr>
<tr>
<td>P38</td>
<td>Designing a Smart City Internet of Things Platform with Microservice</td>
<td>A. Kryloviy; M. Jahn; E. Patti</td>
<td>2015</td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P39</td>
<td>Open Issues in Scheduling Microservices in the Cloud</td>
<td>M. Fazio; A. Celesti; R. Ranjan; C. Liu; L. Chen; M. Villari</td>
<td>2016</td>
</tr>
<tr>
<td>P40</td>
<td>Migrating web applications to clouds with microservice architectures</td>
<td>J. Lin; L. C. Lin; S. Huang</td>
<td>2016</td>
</tr>
<tr>
<td>P41</td>
<td>The ENTICE approach to decompose monolithic services into microservices</td>
<td>G. Keckemeti; A. C. Marosi; A. Kertesz</td>
<td>2015</td>
</tr>
<tr>
<td>P42</td>
<td>Gremlin: Systematic Resilience Testing of Microservices</td>
<td>V. Heerhadi; S. Rajagopalan; H. Jamjoom; M. K. Reiter; V. Sekar</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>Architectures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P45</td>
<td>SeCos: Web of Things platform based on a microservices architecture and</td>
<td>Zeiner H.; Goller M.; Exposito Jiménez V.J.; Salmofero H.; Haas W.</td>
<td>2016</td>
</tr>
<tr>
<td></td>
<td>support of time-awareness</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P46</td>
<td>Multi cloud deployment with containers</td>
<td>Jambunathan B.; Kalpana Y.</td>
<td>2016</td>
</tr>
</tbody>
</table>

(continued on next page)
<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Authors</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>P54</td>
<td>A methodology and tool support for widget-based web application development</td>
<td>Nicolaescu P, Klammar R.</td>
<td>2015</td>
</tr>
<tr>
<td>P55</td>
<td>Learning-based testing of distributed microservice architectures: Correctness and fault injection</td>
<td>Menke K, Nycander P.</td>
<td>2015</td>
</tr>
<tr>
<td>P56</td>
<td>Microservices validation: Methodology and implementation</td>
<td>Sazchenko D, Radchenko G.</td>
<td>2015</td>
</tr>
<tr>
<td>P57</td>
<td>Automated Deployment of a Microservice-based Monitoring Infrastructure</td>
<td>Ciufoletti A.</td>
<td>2015</td>
</tr>
<tr>
<td>P58</td>
<td>An ecosystem of user-facing microservices supported by semantic models</td>
<td>Versteden A, Pauwels E, Papantonious A.</td>
<td>2015</td>
</tr>
<tr>
<td>P60</td>
<td>mCloud: Enabling end-user mobile service creation</td>
<td>Davies M, Carrez F, Heinila J, Fensel A, Narganes M, Danado J.C.S.</td>
<td>2011</td>
</tr>
<tr>
<td>P64</td>
<td>Model-driven Generation of Microservice Architectures for Benchmarking Performance and Resilience Engineering Approaches</td>
<td>Thomas F, Dullmann and Andrésen Hoon</td>
<td>2017</td>
</tr>
<tr>
<td>P65</td>
<td>Towards Effective Virtualization of Intrusion Detection Systems</td>
<td>Nuyin Zhang and Hongda Li and Hongsin Hu and Younghee Park</td>
<td>2017</td>
</tr>
<tr>
<td>P66</td>
<td>Publishing Linked Data Through Semantic Microservices Composition</td>
<td>Ivan Salvadori and Alexis Huf and Ronaldo dos Santos Mello and Frank Siqueira</td>
<td>2016</td>
</tr>
<tr>
<td>P67</td>
<td>An Architecture to Automate Performance Tests on Microservices</td>
<td>André; de Camargo and Ivan Salvadori and Ronaldo dos Santos Mello and Frank Siqueira</td>
<td>2016</td>
</tr>
<tr>
<td>P68</td>
<td>Design and implementation of a decentralized message bus for microservices</td>
<td>Rookarinarat, Pakorn and Tewannaapt, Yawadee</td>
<td>2016</td>
</tr>
<tr>
<td>P69</td>
<td>Telecom strategies for service discovery in microservice environments</td>
<td>C. Rotter; J. Ills; G. Nyiri; L. farkas; G. Csárdi; G. Huszty</td>
<td>2017</td>
</tr>
<tr>
<td>P70</td>
<td>A VNF-as-a-service design through micro-services disassembling the IMS</td>
<td>A. Boubendir; E. Bertin; N. Simoni</td>
<td>2016</td>
</tr>
<tr>
<td>P71</td>
<td>Requirements Reconciliation for Scalable and Secure Microservice (De)composition</td>
<td>M. Ahmadvand; A. Ibrahim</td>
<td>2016</td>
</tr>
<tr>
<td>P72</td>
<td>Towards the understanding and evolution of monolithic applications as microservices</td>
<td>D. Escober; D. Cárdenas; R. Amarillo; E. Castro; K. García; C. Parra; R. Casallas</td>
<td>2016</td>
</tr>
<tr>
<td>P73</td>
<td>A scalable routing mechanism for stateful microservices</td>
<td>N. H. do; T. van Do; X. thi Tran; L. farkas; C. Rotter</td>
<td>2016</td>
</tr>
<tr>
<td>P74</td>
<td>A new efficient distributed computing middleware based on cloud micro-services for HPC</td>
<td>F. Z. Benchara; M. Youssfi; O. Bouattane; H. Ouajji</td>
<td>2016</td>
</tr>
<tr>
<td>P75</td>
<td>Efficiency Analysis of Provisioning Microservices</td>
<td>H. Khazaeei; C. Barna; N. Beigi-Mohammadi; M. Litou</td>
<td>2016</td>
</tr>
<tr>
<td>P76</td>
<td>A microservice based reference architecture model in the context of enterprise architecture</td>
<td>Yale Yu; H. Silveira; M. Sundaram</td>
<td>2016</td>
</tr>
<tr>
<td>P77</td>
<td>Reflections on SOA and Microservices</td>
<td>Z. Xiao; I. Wijegunaratne; X. Qiang</td>
<td>2016</td>
</tr>
<tr>
<td>P78</td>
<td>An open IoT framework based on microservices architecture</td>
<td>L. Sun; Y. Li; R. A. Memon</td>
<td>2017</td>
</tr>
<tr>
<td>P79</td>
<td>Modelling and Managing Deployment Costs of Microservice-Based Cloud Applications</td>
<td>P. Leitner; J. Cino; E. Stockli</td>
<td>2016</td>
</tr>
<tr>
<td>P80</td>
<td>The evolution of distributed systems towards microservices architecture</td>
<td>T. Salah; M. Jamal Zemerly; Chan Yeob Yeun; M. Al-Qutayri; Y. Al-Hammadi</td>
<td>2016</td>
</tr>
<tr>
<td>P82</td>
<td>Workload-Based Clustering of Coherent Feature Sets in Microservice Architectures</td>
<td>Sander Klock, Jan Martijn E. M. Van Der Werf, Jan Pieter Guelen, Slinger Jansen</td>
<td>2017</td>
</tr>
<tr>
<td>P83</td>
<td>Microservice Architectures for Scalability, Agility and Reliability in E-Commerce</td>
<td>Wilhelm Hasselbring and Guido Steinacker</td>
<td>2017</td>
</tr>
<tr>
<td>P84</td>
<td>Differences Between Model-driven Development of Service-oriented and Microservice Architecture</td>
<td>F. Rademacher, S. Sachweh and A. Zündorf</td>
<td>2017</td>
</tr>
<tr>
<td>P86</td>
<td>Decision Guidance Models for Microservice Monitoring</td>
<td>S. Haselbock and R. Weinreich</td>
<td>2017</td>
</tr>
<tr>
<td>P87</td>
<td>From monolith to microservices - Lessons learned on an industrial migration to a Web Oriented Architecture</td>
<td>J. Gougouix and D. Tamzalit</td>
<td>2017</td>
</tr>
<tr>
<td>P88</td>
<td>A Dashboard for Microservice Monitoring and Management</td>
<td>B. Mayer and R. Weinreich</td>
<td>2017</td>
</tr>
<tr>
<td>P89</td>
<td>Self-managing cloud-native applications: Design, implementation, and experience</td>
<td>Giovanni Toffetti and Sandro Brunner and Martin Blochlinger and Florian Dudouet and Andrew Edmonds</td>
<td>2017</td>
</tr>
<tr>
<td>P90</td>
<td>The IPOD demo system: A scalable architecture of microservices for reproducible research</td>
<td>Árevalo M, Escobar C, Monasse F, Monzón N, Colom M.</td>
<td>2017</td>
</tr>
<tr>
<td>P91</td>
<td>Microflows: Automated planning and enactment of dynamic workflows comprising semantically-annotated microservices</td>
<td>Oberhauser R.</td>
<td>2017</td>
</tr>
<tr>
<td>P92</td>
<td>Continuous software engineering: A microservices architecture perspective</td>
<td>O’Connor R.V., Elger P., Clarke P.M.</td>
<td>2017</td>
</tr>
<tr>
<td>P93</td>
<td>A microservice architecture for the Intranet of Things and energy in smart buildings</td>
<td>Bao K, Mauser I, Kochanneck S, Xu H, Schmeck H.</td>
<td>2017</td>
</tr>
<tr>
<td>P94</td>
<td>Domain Driven Design and Provision of Micro-services to build Emerging Learning Systems</td>
<td>Khemaja M.</td>
<td>2016</td>
</tr>
<tr>
<td>P95</td>
<td>Cloudware: An emerging software paradigm for cloud computing</td>
<td>D. Guo; W. Wang; G. Zeng; Z. Wei</td>
<td>2016</td>
</tr>
<tr>
<td>P97</td>
<td>Incremental integration of microservices in cloud applications</td>
<td>Zuniga-Prisot, Miguel and Insfran, Emilio and Abrahao, Silvia and Cano-Genoves, Carlos</td>
<td>2016</td>
</tr>
</tbody>
</table>
Table A.1 (continued)

<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Authors</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>P99</td>
<td>A Service-Oriented Approach to Crowdsourcing for Accessible Smart Mobility Scenarios</td>
<td>Miriti S., Prandi C., Salomoni P., Callegati F., Melis A., Prandini M.</td>
<td>2016</td>
</tr>
<tr>
<td>P100</td>
<td>On micro-services architecture</td>
<td>Namiot, Dmitry and Sneepe-Sneeppe, Manfred</td>
<td>2014</td>
</tr>
<tr>
<td>P101</td>
<td>Towards a Technique for Extracting Microservices from Monolithic Enterprise Systems</td>
<td>Levkovitz, Alessandra and Terra, Ricardo y Valente, Marco Tulio</td>
<td>2016</td>
</tr>
<tr>
<td>P102</td>
<td>A dynamic deployment method of micro service oriented to SLA</td>
<td>ZL Ji, Y Liu</td>
<td>2016</td>
</tr>
<tr>
<td>P103</td>
<td>InterSci/ A Scalable Microservice-based Open Source Platform for Smart Cities</td>
<td>Arthur de M. Del Esposte, Fabio Kon, Fabio M. Costa and Nelson Lago</td>
<td>2017</td>
</tr>
</tbody>
</table>

References


Shalom, N., 2017. Building large scale services with microservices white paper. URL: https://cloudify.com/whitepaper/microservices-orchestration-large-scale-services/.


Paolo Di Francesco is a PhD student in Computer Science at Gran Sasso Science Institute, L’Aquila, Italy. His research interests are software engineering, software architecture, and model-driven engineering. His PhD research is focused on microservice-based architectures, where he investigates techniques for architecture recovery, architecture modeling, and migration of legacy applications towards microservices. In 2014, he received his master degree in Computer Science from L’Aquila University (Italy) and Malaralens Högskola (Sweden), as part of the Global Software Engineering European Master Programme (GSEEM). After he completed his master studies and before pursuing his PhD, he was architect and developer for an industrial company, and then an IT consultant freelancer. In 2015 he was awarded a research grant from the University of L’Aquila. More information is available at http://www.paolodifrancesco.com.
Patricia Lago is Full Professor in software engineering at the Vrije Universiteit Amsterdam, where she leads the Software and Services (S2) research group in the Computer Science Department. Her research is in software architecture and software quality with a special emphasis on sustainability. She has a PhD in Control and Computer Engineering from Politecnico di Torino and a Master in Computer Science from the University of Pisa, both in Italy. She is initiator and coordinator of the Computer Science Master Track in Software Engineering and Green IT, and co-founder of the Green Lab, a place where researchers, students and companies collaborate to measure the energy footprint of software solutions and the impact on software quality. She is member of the Steering Committees of IEEE ICSA, ECSA and the ICT4S conference series, member of the IFIP 2.10 Working group on Software Architecture, the IFIP 2.14 Working group on Services-based Systems, and the Dutch Knowledge Network on Green Software. More information is available at www.cs.vu.nl/~patricia.

Ivano Malavolta is Assistant Professor at the Vrije Universiteit Amsterdam, the Netherlands. His research focuses on data-driven software engineering, software engineering for mobile development, software architecture, model-driven engineering (MDE), and robotics. He is applying empirical methods to assess practices and trends in the field of software engineering. He is program committee member and reviewer of international conferences and journals in his fields of interest. He authored more than 80 papers in international journals and peer-reviewed international conferences proceedings. He received a PhD in computer science from the University of L’Aquila in 2012. He is a member of ACM and IEEE, Amsterdam Data Science, and VERSEN. More information is available at http://www.ivanomalavolta.com.