Testing Microservices Architecture-Based Applications: A Systematic Mapping Study

Conference Paper · October 2020

DOI: 10.1109/APSEC51365.2020.00020

4 authors:

Muhammad Waseem
Wuhan University
30 PUBLICATIONS **156** CITATIONS
SEE PROFILE

Peng Liang
Wuhan University
205 PUBLICATIONS **2,839** CITATIONS
SEE PROFILE

Gastón Márquez
Universidad Técnica Federico Santa María (sede Concepción)
43 PUBLICATIONS **350** CITATIONS
SEE PROFILE

Amleto Di Salle
Università degli Studi dell'Aquila
43 PUBLICATIONS **447** CITATIONS
SEE PROFILE

Testing Microservices Architecture-Based Applications: A Systematic Mapping Study

Muhammad Waseem and Peng Liang*
School of Computer Science
Wuhan University
Wuhan, China
{m.waseem, liangp}@whu.edu.cn

Gastón Márquez
School of Informatics Engineering
University of Valparaíso
Valparaíso, Chile
gaston.marquez@uv.cl

Amleto Di Salle
Department of Information Engineering, Computer Science and Mathematics
University of L'Aquila, L'Aquila, Italy
amleto.disalle@univaq.it

***Abstract*—Microservices is an architectural style that provides several benefits to develop applications as small, independent, and modular services. Building Microservices Architecture (MSA)-based applications is immensely supported by using software testing fundamentals. With the increasing interest in the development of MSA-based applications, it is important to systematically identify, analyze, and classify the publication trends, research themes, approaches, tools, and challenges in the context of testing MSA-based applications. The search yielded 2,481 articles, and 33 articles were finally selected as the primary studies with snowballing. The key findings are that (i) 5 research themes characterize testing approaches in MSA-based applications; (ii) integration and unit testing are the most popular testing approaches; and (iii) addressing the challenges in automated and inter-communication testing is gaining the interest of the community. Additionally, it emerges that there is a lack of dedicated tools to support testing for MSA-based applications, and the reasons and solutions behind the challenges in testing MSA-based applications need to be further explored.**

***Index Terms*—Microservices, Microservices Architecture based Application, Testing, Systematic Mapping Study**

I. Introduction

Microservices is a style of architecture in which an extensive application is built as a set of modular components or services. Each module supports a specific task or business objective and uses a well-defined and straightforward interface, such as an Application Programming Interface (API), to communicate with other sets of services [1]. In recent years, microservices have become more popular, and companies prefer to move from a monolithic architecture to Microservices Architecture (MSA). Typically, hundreds to thousands of microservices may be included in large scale MSA-based systems. For instance, more than 500 microservices of Netflix handle approximately 2 billion API edge requests every day [2], whereas Tencent's WeChat system is composed of more than 3,000 services running over 20,000 machines [3].

MSA is emerged from Service-Oriented Architecture (SOA), with an emphasise on the compensation of small services, implementation of DevOps and agile [4]. However, they vary significantly in terms of service characteristics, such as service granularity, service communication, and component sharing [5]. For instance, MSA contains hundreds of fine-grained services working together, and they evolve very quickly. SOA has extensive and more modular and stable services. MSA communicates through an API layer, whereas SOA can communicate via an API (specified as a WSDL interface) and Enterprise Services Bus (ESB). MSA provides better scalability, decoupling, and control over application implementation (e.g., development, testing, deployment) than SOA [5]. These fundamental differences raise some challenges in testing strategies specific to MSA-based applications. Typically, MSA is used for a single, sophisticated application, as SOA is better suitable for systems integration [6].

Software testing is composed of dynamic verifications that evaluate if a system provides expected behavior on a finite set of test cases, suitably selected from the usually infinite execution domain [7]. The tests are structured in a series of activities that have a purpose of finding the possibility of implementation, quality or usability failures of a program or system. The number of services, inter-communication processes, dependencies, instances, network communication, and other variables influence testing methodologies in MSA-based applications. MSA-based applications pose significant challenges for testing due to their complex nature and dynamics behavior. During the testing process, it needs to understand the concurrent behaviors of the various microservices and interactions between them [8]. The academia and industry discuss many approaches for managing the testing complexity in MSA-based applications. These approaches cover multiple independently deployable components as well as check if the system remains correct despite having multiple development teams.

The growing interest of microservices research and development in academia and industry alike has led to an increased number of works describing approaches, challenges, tools, and practices related to this field. Ample evidence of these trends can be found in existing secondary studies (e.g., [9][10][11][12][13]). However, a dedicated effort in systematically analyzing and synthesizing the literature related to testing approaches for MSA-based applications is still non-existent.

In this paper, we present a Systematic Mapping Study (SMS) about the use of testing approaches in the context of MSA. Out of 2,481 total studies retrieved, we obtained 33 primary studies. We examined in detail by discussing

* Corresponding author

key findings of research themes, testing approaches, and challenges. The main **contribution** of our work is a state-of-the-art about testing approaches in MSA-based applications.

The rest of this paper is structured as follows: Section II describes the mapping study design; Section III shows the results; Section IV discusses the key findings; Section V presents the threats to validity; Section VI discusses related works; Section VII concludes the study with recommendations for future areas of research.

II. Mapping Study Design

We used the guidelines proposed by Petersen et al. [14] for conducting SMSs, and the strategies presented by Kitchenham et al. [15] for Systematic Literature Reviews (SLR).

A. Research Goal and Questions

The goal of this SMS is to *analyze the peer-reviewed literature concerning publication trends, research themes, approaches, and challenges in the context of testing MSA-based applications*. Therefore, we derived three Research Questions (RQs), which are:

RQ1: *What are the existing research themes on testing MSA-based applications and how can they be classified and mapped?*

Rationale: This RQ aims to establish the foundation for systematic analysis of the existing research on testing MSA-based applications through a taxonomy of research themes and sub-themes. In this regard, we applied the approach proposed by Braun et al. [16] to identify the main research themes related to the primary studies.

RQ2: *What testing approaches and tools have been proposed and used for MSA-based applications?*

Rationale: Testing approaches refer to the strategies, methodologies, and techniques used to test an application to ensure that it behaves and looks as expected [17]. Consequently, this RQ aims to collect strategies, methodologies, techniques, tools, and other proposals that are used to test MSA-based applications.

RQ3: *Which testing-related challenges have been reported in the primary studies over the years?*

Rationale: The distributed nature of microservices that are built around business capabilities, independently deployable and packaged, poses significant challenges to the testing of MSA-based applications. By answering this RQ, we aim to highlight the challenges reported in the primary studies concerning testing of MSA-based applications. Furthermore, with this RQ, we want to present the evolution of the testing challenges over the years.

B. Study Search and Selection Process

The search and selection process is divided into two phases. Phase I corresponds to the primary search, and Phase II is related to the snowballing process (see Figure 1). In the following, we further describe the detail of each phase.

Phase I - Primary Search: The existing SMS/SLRs identified that the first primary study regarding MSA that was included in any SMS [12] was published in 2008. Therefore, we executed the search string from January 2008 to November 2019 in seven major electronic databases (see Table I). Then, we selected relevant studies by following the steps below:

TABLE I: Selected databases and search string

Search string	
((microservice OR micro service* OR micro-service* OR microservices architect* OR microservices design) AND test*)*	
Databases	
Database	**Targeted search area**
ACM Digital Library	Paper title, abstract
IEEE Explore	Paper title, keywords, abstract
Springer Link	Paper title, abstract
Science Direct	Paper title, keywords, abstract
Wiley InterScience	Paper title, abstract
EI Compendex	Paper title, abstract
ISI Web of Science	Paper title, keywords, abstract

Step 1 - Studies extraction: We executed the search string on selected databases aiming to obtain the information on study title, authors list, publication year, study venue, publication type, and summary/abstract.

Step 2 - Screening studies through titles: Before starting the formal screening by reading the title of studies, we removed duplicate studies by sorting them in ascending order and, subsequently, removed duplicate titles. Then, two researchers of our team divided the remaining studies equally and started screening by reading their titles and keywords (independently). According to the available evidence about our research context (i.e., testing of MSA-based applications) in studies' titles and keywords, we removed several hundred irrelevant studies. Any disagreements about the results of screening studies were discussed among all the authors to get a consensus.

Step 3 – Screening studies through reading abstracts: One researcher reviewed the pool of studies compiled after Step 2. Then, the same researcher read the abstract of every study and labeled it as "relevant", "irrelevant", or "doubtful". In order to reduce bias on the labeling results, another researcher performed the same procedure. In this step, both researchers did not find any study where they could not reach on a final consensus.

Step 4 - Applying inclusion and exclusion criteria: Finally, we executed the inclusion and exclusion criteria (see Table II) in order to get the primary studies. We did not include studies related to SOA since MSA and SOA have fundamental differences (as stated in Section I), and we were looking for distinct outcomes for microservice testing. Moreover, reviews on testing SOA are already available (e.g., [18]). Again in this step, any disagreements on the selected studies were discussed among all the authors to achieve a consensus.

Phase II - Snowballing: Further relevant studies are identified by applying the snowballing procedure [19]. We performed both backward and forward snowballing (i.e., references, citations) procedure by taking 28 selected studies (output of Step 4) as an initial dataset. After this step (5 studies), we obtained the final set of included studies (33) for data extraction to answer the RQs.

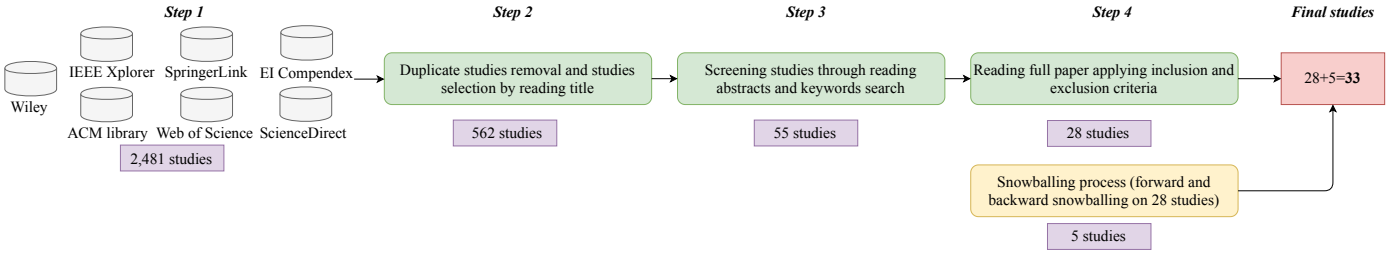

Fig. 1: Study search and selection process

TABLE II: Inclusion and exclusion criteria

Selection Criteria	Inclusion Criteria	Exclusion Criteria
Language	English	Non-English
Study Type	Primary studies	Secondary studies
	Peer reviewed journals articles, book chapters, conference papers, workshop, and symposiums papers	Blog, webpages, videos, white papers, technical reports, non-pee review studies (the so-called grey literature).
Study Focus	Studies that explicitly discuss the testing for MSA-based applications	Studies that discuss the testing for SOA and monolithic applications
	Studies that could provide the information regarding data items D8, D9, and D10	Studies that do not provide the information regarding data items D8, D9, and D10
Study Duration	A study published between 2008 and November 2019	A study published before 2008 or after November 2019

C. Search String

The search string used in this study generally represents the focus of our research (i.e., microservices and testing). However, we firmed the search string by considering the research objective, RQs, writing style of microservices (e.g., microservice, micro-services), and synonyms for architecture (e.g., design). Because the keyword "testing" itself is self-explanatory, therefore, we did not use any other alternative word for it. We also included the wildcard with keywords to maximize the search results (see Table I). The search string was iteratively improved through pilot searches to mitigate the risk of missing relevant studies.

D. Data Extraction Form

We designed the data extraction form according to the data items shown in Table III. The data items (D1 to D7) are used to present an overview of the selected studies. Concerning data item D8, we applied Thematic Analysis [16] to identify the main research themes concerning the primary studies, which is a method for systematically identifying, organizing, and offering insight into patterns of meaning (themes) across an extracted dataset, and is composed of six steps:

1) *Familiarizing with the data*: This step aims to read and transcribe data regarding data item D8.
2) *Generating initial codes*: This step systematically produces the representative codes of the selected studies for defining research themes.
3) *Searching for themes*: In this step, generated codes are analyzed and classified as potential themes.
4) *Reviewing themes*: This step checks the suitability of the themes concerning the codes and the dataset in order to create a thematic "map".
5) *Defining and naming themes*: This step aims to define, refine and characterize all themes using precise and clear names.
6) *Producing the report*: This last step aims to refine the identified themes and their corresponding characteristics.

TABLE III: Data extraction items

Code	Data Item	Description	RQ
D1	Index	The ID of the study	Demographics
D2	Title	The title of the study	
D3	Author(s) list	The full name of the authors	
D4	Year	Publication year of the study	
D5	Venue	The name of the publishing venue	
D6	Publication type	Journal, conference, workshop, bookchapter, and technical report	
D7	Research Type	Case study, survey, experiments, validation research, solution proposal, etc	
D8	Research theme	Execution of guidelines proposed by Braun et al. [16]	RQ1
D9	Testing approaches	The approaches (e.g., unit testing, integration testing, consumer testing) reported in the study to test MSA-based applications	RQ2
D10	Testing challenges	The challenges reported in the study about testing MSA-based applications	RQ3

The process of obtaining research themes executed in this SMS is provided in Figure 2. Two researchers participated in the process in order to reduce bias. The most important activity of this process was brainstorming sessions that were mainly conducted while reviewing, defining, and naming the research themes. In these sessions, both researchers discussed and validated the research themes found.

Concerning data item D9 (testing approaches) and D10 (testing challenges), we used open coding and constant comparison techniques from Grounded Theory [20] to analyze the qualitative data extracted from the selected studies. Finally,

we created a replication package [21] containing the detailed protocol of our SMS with all the important data (e.g., primary studies, publication venues, research types) related to the SMS.

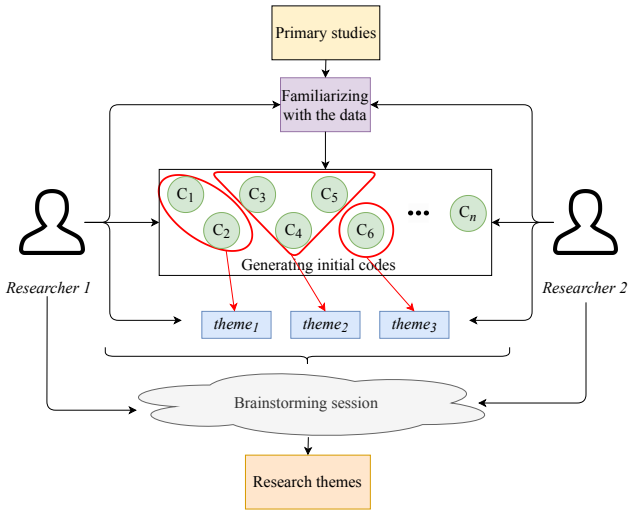

Fig. 2: A research themes classification procedure from the primary studies

III. Results

A. Demographics

We obtained a total of 33 primary studies published from 2015 to November 2019 (see Figure 3). It is clear to see a significant increase in the number of publications related to testing of MSA-based applications from 2017 to 2019.

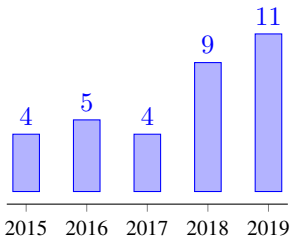

Fig. 3: Studies distribution over publication years

On the other hand, the result shows that the 33 selected studies were published in 29 venues, including 21 conferences (precisely four conferences, SOSE, SCC, COMPSAC, and NetSoft, published two studies, respectively), 5 journals, and 3 workshops. The number of venues indicates that research on testing MSA-based applications is gaining popularity across multiple venues.

In order to classify the primary studies from a research type perspective, we used the proposal of Wieringa et al. [22] for articles classification. Figure 4 shows that 48.5% of the primary studies are related to the proposal of solution. This means that these primary studies proposed a solution technique and argued for its relevance, without a full-blown validation. The rest of the categories that characterize the primary studies indicate that (i) some studies (7 out of 33, 21.2%) investigated the properties of a solution proposal that has not yet been implemented (Validation research), (ii) some other studies (7 out of 33, 21.2%) discuss the investigation of a problem or an implementation of a technique in practice (Evaluation research), and (iii) some studies (3 out of 33, 9.1%) contain the authors' opinions about what is wrong or right about something (Opinion paper).

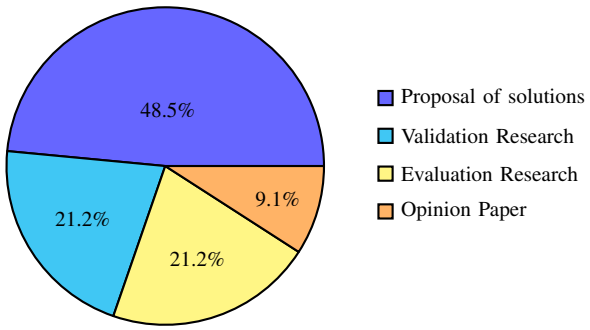

Fig. 4: Research type of the primary studies

We also classify the primary studies from the research method perspective (see Figure 5). The results show that 45.5% (15 out of 33) of the primary studies used case studies, 27.3% (9 out of 33) primary studies used experiments, whereas only 6.1% (2 out of 33) of studies have used the surveys. On the other hand, A significant number of studies (7 out of 33, 27.3%) did not describe any research method.

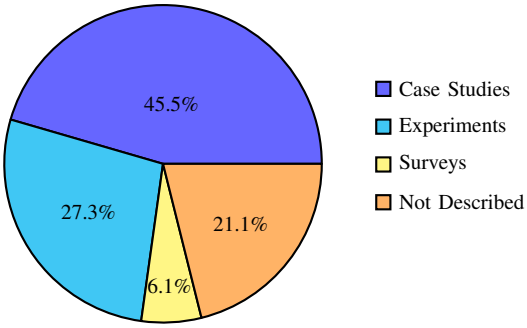

Fig. 5: Research methods used in primary studies

B. RQ1: Research Themes

We identified five main themes that characterize the primary studies such as automated testing (10 out of 33, 30.3%), architecture (8 out of 33, 24.3%), DevOps and CI (8 out of 33, 24.3%), performance (3 out of 33, 9.1%), and model-based testing (3 out of 33, 9.1%). Subsequently, Table IV describes the sub-themes and their corresponding main points related to testing MSA-based applications. Note that one study may have multiple themes, but is only classified into one theme, for example, Study [6] and Study [23] also have the theme "Automated testing" besides "Performance" and "DevOps and CI", respectively. In the following, we further describe each research theme.

TABLE IV: A classification of research themes on testing MSA-based applications from the primary studies

Themes	Subthemes	Key points	Study
Automated testing	Acceptance testing	A formal specification-based approach to derive test cases (e.g., acceptance test cases) for automated microservice testing	[24]
		Automated acceptance testing for microservices in behavior-driven development	[25]
	Functional testing	Test automation for functional testing of micro services and service architectures	[26]
		Usage of TTCN-3 Service Profiling Format (TSPF) approach for functional testing of MSA-based virtualized network	[27]
	Learning-based testing	Automated black-box testing for MSA that combines the machine learning with model checking	[28]
	Parallel testing	Framework for parallel test execution for web-based applications (e.g., microservices and back-end web services)	[29]
	Automation framework	Full stack microservices framework that automatically create test configurations to ease testing workflows for integration, load, and end-to-end testing	[30]
	Tools support	Open source tools that allow automation of testing for MSA-based applications	[31]
	Migration test case extraction	Test cases extraction methodology to identify the impact of migration from legacy architecture to MSA	[32]
	Test cases retrieval scheme	Service dependency graph based scheme to analyze, test, and reuse microservices	[33]
Architecture	System level test cases	System level test cases for RESTful web services (e.g., microservices) by using evolutionary algorithms	[34]
	Test containers	Testing microservices inside containers	[35]
	Reliability	Testing method for on-demand reliability estimation of MSA-based applications	[36]
	Resilience	Testing the resilience of microservices in production infrastructure	[37]
	Functional testing	Testing of MSA-based advanced driver assistance system (ADAS) functions	[38]
	MSA evaluation	High-level evaluation of the MSA for automotive use cases	[38]
		Fault analysis and debugging of MSA-based applications	[39]
	Component and integration testing	Approach for component and integration testing of MSA-based applications	[40]
	Consumer-driven contract testing	Testing of MSA-based applications through consumer-driven contract tests	[41]
DevOps and CI	Continuous testing	Continuous testing of MSA-based applications in production environments	[42][43]
		Service and end user application testing in production with continuous delivery	[44]
		Continuous testing of MSA-based peer-to-peer distributed software systems	[45]
		Continuous testing of MSA-based Open Network Operating System (ONOS)	[23]
		Continuous testing of MSA-based Bifrost middleware	[46]
	Continuous delivery	A continue delivery pipeline for design and development of MSA-based applications	[42][43]
		Regression testing of MSA-based applications in continuous delivery	[47]
		Black-box and white-box testing of MSA-based applications in DevOps	[48]
Performance	Performance testing	Evaluation of the performance that each microservice can deliver	[6]
		Method of performance testing with Kubemark	[49]
	Performance comparison	Tested and compared monoliths and microservices applications' performance in NFV	[50]
Model-based testing	Dependency graph	Graph-based microservice analysis and testing approach	[51]
	Petri nets	Petri nets as the basis of model-based testing of microservices	[52]
	Load testing	Generation of session-based workload models for load testing of microservices	[53]

a) ***Automated testing****:* This theme covers the primary studies which discuss specific tests via automated testing. As an example, Quenum and Aknine present an automated testing approach based on a formal specification and intelligent agents (i.e., LASTA automated testing) to derive test cases (e.g., unit and acceptance tests) for microservices [24]. Similarly, Shang et al. introduce a graph-based and scenario-driven scheme to analyze, test, and reuse microservices [33]. This approach enables the automatic retrieval of test cases required to deal with microservice changes.

b) ***Architecture****:* In this research theme, the discussion of the primary studies focuses on using or considering architecture artifacts (e.g., architecture components, quality attributes) to test MSA-based applications. More precisely, the primary studies present testing approaches that use MSA design components (e.g., services, service interface), application decomposition strategies, and microservices communication methods (e.g., synchronous or asynchronous protocols) to create test cases for testing MSA-based applications. Moreover, several studies (e.g. [36], [37], [38]) describe how to test and evaluate the reliability, resilience, and architecture of MSA-based applications in this theme.

c) ***DevOps and CI****:* DevOps has a range of practices (e.g., Continuous Integration (CI), Continuous Delivery (CD), testing, deployment) intended to deliver reliable software systems by encouraging close cooperation between development and operational staff. This theme includes the studies that report the testing approaches and tools used for MSA-based applications in DevOps or CI context. Test automation is a key factor in succeeding with DevOps. For example, Kargar and Hanifizade proposed an automated method to support the regression testing of microservices in continuous delivery [47]. Marcel and Christos developed and tested MSA-based applications for Open Network Operating System (ONOS) by establishing the CI environment [23]. They also proposed a trial topology used to evaluate ONOS applications.

d) ***Performance****:* This theme deals with the quantitative behavioral aspects of testing. The result shows that the primary studies related to this topic focus on performance testing, mostly in the production phase of MSA-based application development. For instance, Camargo et al. present an approach to evaluate individual microservice performance by integrating test specifications [6]. Moreover, Sharma et al. tested and compared the performance between monolithic and microservices applications in the field of Network Function Virtualization (NFV) by using the analytic model and implementing test-bed experiments [50].

*e) **Model-based testing (MBT):*** This theme gathers those primary studies that discuss the MBT approaches for MSA-based applications. For example, Camilli et al. present a formal framework based on Petri net models that is applicable in the context of microservices testing [52]. Schulz et al. introduce an MBT approach to generate workload models for load testing of one or more specified microservice(s) [53].

Key finding of RQ1:
Researchers are putting more effort in proposing and implementing testing approaches for MSA-based applications in the context of automated testing, the use of architectural artifacts, and DevOps and CI.

C. RQ2: Testing Approaches and Tools

We identified 39 testing approaches (see Figure 6 and [21]) and 5 tools (see Table V) for MSA-based applications from the selected studies. The main testing approach mentioned in primary studies is integration testing (12 out of 33, 36.4%). This type of test aims to verify the correct assembly between the different components. In particular, it verifies the proper interaction through their interface, both internal and external, to cover the established functionality and adjust to the non-functional requirements specified in the corresponding verifications. Subsequently, the testing approach, which is also highly mentioned in primary studies, is unit testing (9 out of 33, 27.3%). This test consists of isolating part of codes and verifying if it works correctly focused on validating the behavior of the service and their corresponding logic. In the context of microservices, primary studies agree that unit testing is widely used to test the business logic of each microservice. Mainly, unit testing tests each service as an independent module, and generally, they are used as white-box tests (as opposed to integration tests considered as a black box). Note that we present the testing approaches collected from the selected studies, and these testing approaches (see Figure 6) are not orthogonal and may contain each other. For example, unit testing may involve different testing approaches, such as evolutionary algorithm-based testing, grey box testing, and model-based testing [54].

On the other hand, we identified several studies that present novel approaches. The presented approaches and methodologies can be used to support various microservices testing strategies (e.g., acceptance testing, integration testing, contract testing). As an example, intelligent agents and formal specification-based approach for microservices [24], approach to automate acceptance testing for microservices [25], TTCN-3 Service Profiling Format (TSPF) testing approach for microservices [27], methodology for extracting migration test cases by using Impact Data All Used (IDAU) and graph analysis techniques for microservices [32], Graph-based and Scenario-driven Microservice Analysis Retrieval and Testing (GSMART) approach [33], model-based microservices testing approach by using dependency graph and Petri-nets [51][52], and EvoMaster tool for generating system-level test cases for microservices by using evolutionary algorithms (e.g., MIO, WTS, MOSA) [34].

Regarding testing tools, Table V summarizes the tools that are mentioned in the primary studies. We classified the tools based on the testing strategies collected by Clemson [55] for MSA-based applications, which are unit testing, integration testing, component testing, contract testing, and end-to-end testing. We observed that most of the identified tools are not mainly developed to test MSA-based applications, but can be used for this purpose.

Key findings of RQ2:
(i) The most frequently mentioned testing approaches to test MSA-based systems are integration and unit testing, which are also used to test monolith and SOA-based systems. (ii) Several novel testing approaches are proposed to test MSA-based applications. However, these testing approaches are not applied to real-world MSA-based applications. (iii) Lack of dedicated tool support to test MSA-based applications.

D. RQ3: Challenges

The primary studies reported several challenges related to testing of MSA-based applications. Most of the identified challenges are belongs to testing approaches for MSA-based applications. Using Figure 6 as a reference, we summarize the following challenges.

- *Automated testing*: Automated testing means using tools and techniques to execute test cases based on a certain degree of automation to test software. The challenges related to this approach point to evaluating MSA-based applications with a considerable number of microservices. More precisely, challenges lie in automating testing when the deployment is complex (several services deployed), having several types of services in an application, and using the wrong tools.
- *Inter-communication testing*: This approach aims to test communication mechanisms (synchronous or asynchronous) that involve inter-communication between services. We identified that inter-communication between hundreds of microservices poses the challenges for integration and system-level testing due to the complexity shift from the individual microservice to a system [31][38].
- *Faster test feedback*: Manual or automatic testing procedures whose purpose is to provide feedback concerning the performance of an application in early development phases. We found that existing testing framework does not provide reliable test feedback for the systems (e.g., MSA-based applications) that are based on file system access, database fixtures, and network communication [29].
- *Integration testing*: The challenges related to integration testing emerge when combinatorial expansion between microservices interactions exist, and microservices are deployed on multiple platforms [23]. Moreover, analyzing logs across multiple microservices and writing effective integration test cases for them can also be very twitchy and mentally taxing for quality assurance personals [56].
- *Acceptance testing*: Acceptance tests establish the degree of confidence in a system, parts of it, or its non-functional characteristics. However, it is difficult due to the lack of

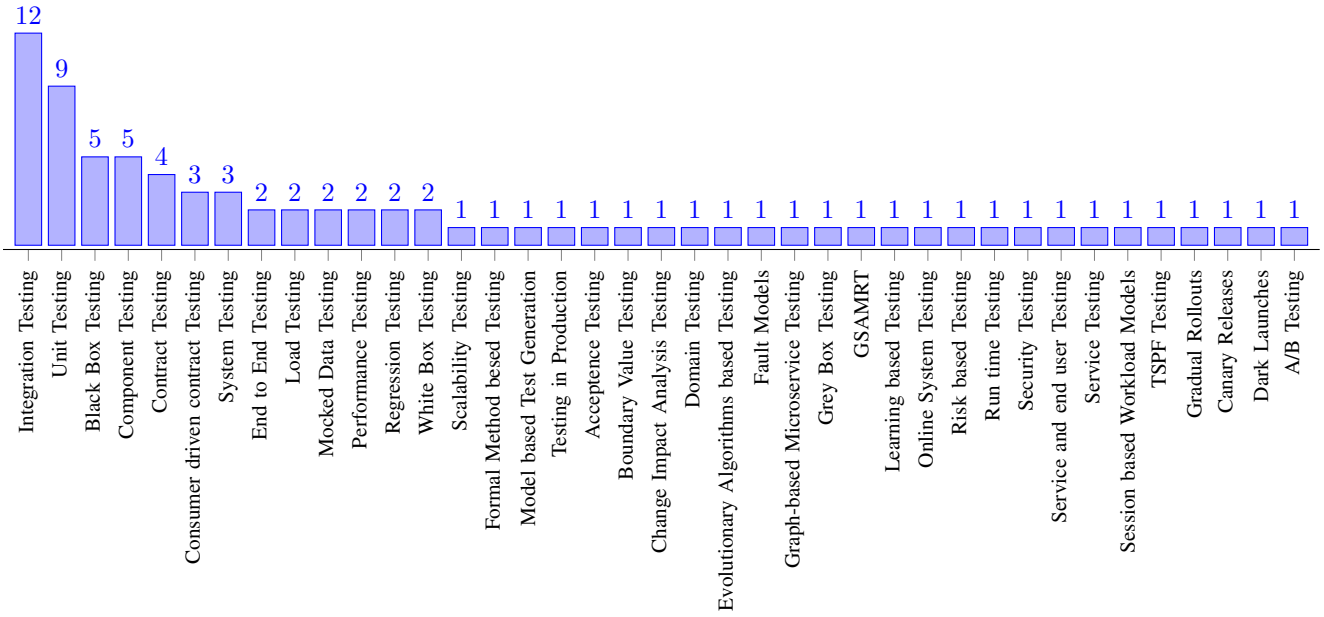

Fig. 6: Testing approaches for MSA-based applications from the primary studies

TABLE V: Testing tools used in the primary studies

Classification	Tool	Description	Usage in testing MSA-based applications
Unit testing	JUnit	Testing framework for unit testing	Microservices functionality
Integration testing	Mockito	Set of libraries that enable mock creation, verification, and stubbing	Function calls
Contract testing	Pact	Contract testing tool	Inter-communication between services
End-to-End testing	Selenium	Framework for testing web applications	Test if MSA-based applications meet specific requirements
	Nightmare	High-level browser automation library	Test APIs and system requirements

tools for supporting the acceptance testing of MSA-based applications in Behavior-Driven Development (BDD) [25]. By overcoming this difficulty, the test execution time can be decreased significantly.

- *Granularity testing*: This testing is focused on testing the specific functionalities of software systems. Granularity testing is challenging because of managing chaining interfaces and increased asynchronicity complexities of MSA-based applications [35].
- *Performance testing*: This testing aims to evaluate system behavior and performance, particularly response time, resource usage, and scalability. Performance testing is challenging for MSA-based applications due to introducing new business requirements in the operational phase, poorly organized MSA, and dependency between dozens of microservices [49].
- *Runtime testing*: This testing aims to find errors, defects, and failures on software at runtime. Generating and executing tests at runtime is challenging because of the high flexibility and evolvability nature of MSA [36]. However, runtime testing is important for reliability estimation of MSA-based applications in their operational phase.

In summary, the challenges mentioned in the primary studies over the years are illustrated in Figure 7. Most of the studies report automated and inter-communication testing as challenges. Automated testing is the challenge that has continuously been reported over the years. This type of testing is attractive for software testing because it opens the possibility to include sophisticated automation techniques to reduce human intervention in software testing. In the context of microservices, automated testing becomes a great ally to test MSA-based applications.

On the other hand, inter-communication testing also brings several challenges. In this regard, the primary studies describe that most challenges are related to verifying if the interactions between microservices follow the contract rules or agreements (i.e., protocols). For this reason, developers have an emerging interest in testing communication protocols (such as AMQP, HTTP/HTTPS, TCP) among microservices.

Key finding of RQ3:
There is a growing interest to address automated testing challenges and inter-communication testing challenges between microservices.

IV. Discussion

RQ1: Research Themes: Systematic identification of the research themes about testing MSA-based applications is provided in Table IV, which shows that the most recurring topics are automated testing (10 studies, 30.3%), architecture (8 studies, 24.3%), and DevOps and CI (8 studies, 24.3%). These results indicate that researchers are not only putting more effort in proposing approaches for automating the

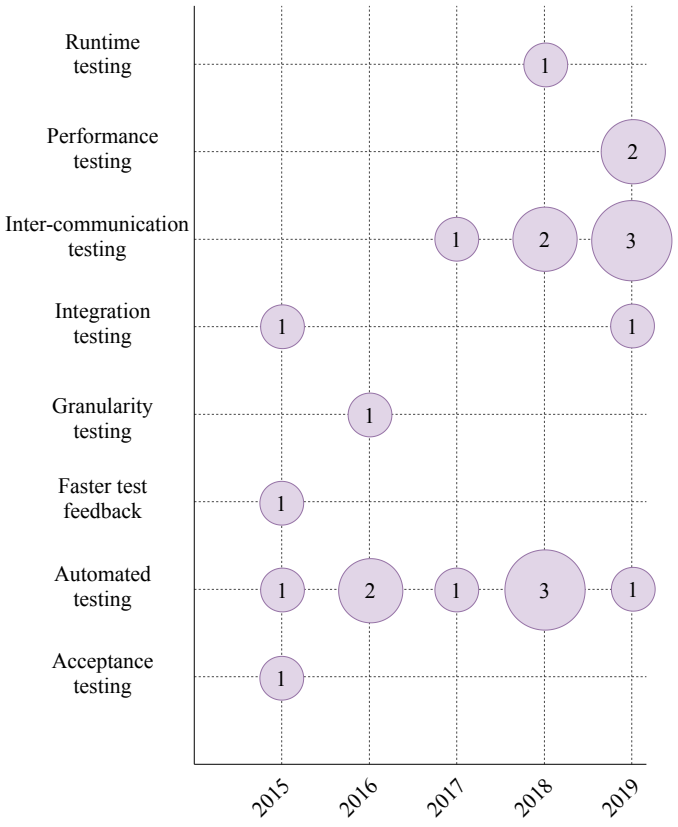

Fig. 7: Challenges in the primary studies

testing process, but they also implement and test MSA-based applications in DevOps. Based on the results presented in Section III-B, our key observations are: (i) model-based testing and several other approaches are proposed to automate microservices testing strategies (e.g., acceptance testing, functional testing). However, most of them are not validated through real-world industrial projects, (ii) only a few studies reported the research regarding certain quality attributes (i.e., performance, reliability, and resilience). An evaluation of the trade-off between quality attributes could be an exciting future research direction. For example, the trade-off between microservices scalability and security, and (iii) several studies (e.g., [28]) reported the development and testing experience of MSA-based applications in DevOps. However, these studies do not provide detailed information regarding the testing process. For instance, how to create and execute unit, integration, regression, and performance test cases for MSA-based applications in DevOps.

On the other side, we found that there is a lack of benchmark programs for the evaluation of MSA testing techniques. We found only one study (i.e., [39]) that present the fault analysis and debugging of MSA-based application by considering the medium-size benchmark MSA-based application. One other study (i.e., [46]) evaluated the live testing strategies (i.e., Canary launch, Dark launch, A/B Test, Gradual rollout) on MSA-based applications. The system (i.e., Bifrost) used to evaluate in this study can also be used as a benchmark program for MSA testing techniques.

RQ2: Testing Approaches and Tools: Regarding testing approaches for MSA-based applications, we found that integration testing (48.5 studies, 51.6%) and unit testing (12 studies, 36.4%) are dominating methods for MSA-based applications (see Figure 6). We made three major observations regarding testing approaches and tools: (i) Integration and unit testing are showing positive results when faced with the problems concerning the complexity of services and deployment. However, they are not enough to thoroughly test an MSA-based application. (ii) There is a lack of evidence from practice about the effectiveness of novel testing approaches (e.g., TSPF) that are reported only in one or two studies (see Figure 6). Hence, the efficacy of these testing approaches for complex real-world MSA-based applications is questionable. (iii) All the identified tools (e.g., JUnit) are general-purpose tools that are not developed to test MSA-based applications dedicatedly.

RQ3: Challenges: We identified the challenges reported in the primary studies over the years in the context of testing MSA-based applications (see Figure 7), which are mainly associated with automated testing, inter-communication testing, faster test feedback, and integration testing. However, we could not find the reasons and solutions behind these challenges in the selected studies, which should be further explored together.

V. Threats to Validity

Threats to *internal validity* represent circumstances that could influence the results obtained from the research. We addressed the following threats: (i) Study selection bias: To mitigate this threat, we have explicitly defined and applied the inclusion and exclusion criteria in Table II. Additionally, we conducted a cross-check validation with all primary studies. (ii) Data extraction bias: Aiming at decreasing data extraction bias and ensure data consistency, we have designed a form for data extraction (see Table III). Two researchers from our team proceeded to extract data from evenly distributed primary studies. Besides, data extraction conflicts are resolved through continuous discussion between all the authors to reduce bias. (iii) Research themes classification bias: In order to reduce bias in the identification of research themes, we used the guidelines described by Braun et al. [16], which describe qualitative methods and techniques to characterize research themes in documents.

The potential threats to *external validity* are related to the degree in which the results of a study can be generalized. We tackled this threat by developing the study protocol [21] that rigorously specifies the whole process of conducting this SMS.

Construct validity is related to taking correct operational measures for collecting the data to study. An inaccurate or incomplete search string can bring threats like excluding relevant papers and including irrelevant papers. To mitigate these threats, we took the following corrective actions: (i) pilot searches to ensure the correctness of the search terms, (ii)

customizing the search string according to a database format, and (iii) executing the search on popular databases.

Threats to the *conclusion validity* directly affect the ability to represent correct conclusions. In order to mitigate this validity threat, we used the guidelines of Kitchenham et al. [15] for executing SLRs and, additionally, discussed the results in several brainstorming sessions. This discussion allowed us to analyze each primary study more thoroughly.

VI. Related Work

Several systematic studies published in recent years aimed at examining microservice growth and status from a different perspective. Alshuqayran et al. [9] conducted an SMS to explore the MSA and their implementation. Their study mainly investigated the architectural challenges (e.g., communication/integration, service discovery, performance, fault tolerance) in the context of microservices. In addition, Taibi et al. [10] and Márquez et al. [13] reported architectural patterns and tactics related to microservices. Their SMS identifies and classify microservices patterns, advantages and disadvantages. The study conducted by Soldani et al. [11] reports pains and gains from industrial grey literature related to the design, development, and operation of MSA-based applications. Similarly, Di Francesco et al. [12] present a comprehensive review of architecting microservices concerning the focus of research, industrial adoption, and publication trends. One of their findings is that work on microservices is still in its early stages, and the combination of industrial and academic authors ' participation is encouraging.

We recently conducted an SMS on MSA in DevOps from several perspectives [57], including the research themes, problems, solutions, MSA patterns, quality attributes. The results of this SMS also indicate that testing of microservices in DevOps needs further attention and investigation.

The secondary studies mentioned above not only discussed the challenges/pains, gains, patterns, and research directions in general about MSA-based applications but also suggested the critical need for further research on testing of MSA-based applications. To the best of our knowledge, there is no work available that systematically analyze, summarize, and classify the testing approaches for MSA-based applications.

VII. Conclusions

This paper presents an SMS regarding testing approaches used in MSA-based applications. From an initial set of 2,481 retrieved articles, we obtained 33 primary studies. The results reveal that during 2019, there was an increase in the number of publications related to testing approaches used in MSA-based applications. In turn, the research themes that characterize testing approaches in the context of microservices are Automated Testing, Architecture, DevOps and CI, Performance, and Model-based Testing. On the other hand, key findings obtained from the results are (i) primary studies reveal novel testing approaches, (ii) integration and unit testings are dominating testing approaches, and (iii) automated testing and inter-communication testing are frequently reported challenges.

Our future work focuses on (i) knowing the industrial reality concerning testing for MSA-based applications through surveys and to what extent the research outcomes have been employed in practice, (ii) extending this SMS by including grey literature to identify the gap between research and practice, and (iii) investigating the general-purpose testing approaches for MSA-based applications.

Acknowledgments

This research work has been supported by the National Key R&D Program of China with Grant No. 2018YFB1402800, Comisión Nacional de Investigación Científica (CONICYT) through grants PCHA/Doctorado Nacional/2016-21161005, Centro Basal CCTVal and by the Italian Ministry of Economy and Finance, Cipe resolution n.135/2012 (INCIPICT), and SISMA national research project funded by the MIUR under the PRIN 2017 program (Contract 201752ENYB).

References

[1] S. Newman, *Building Microservices: Designing Fine-Grained Systems*. O'Reilly Media, Inc., 2015.

[2] SmartBear, "Why you can't talk about microservices without mentioning netflix," 2015. accessed on 2020-04-03.

[3] H. Zhou, M. Chen, Q. Lin, Y. Wang, X. She, S. Liu, R. Gu, B. C. Ooi, and J. Yang, "Overload control for scaling wechat microservices," in *Proc. of the 9th ACM Symp. on Cloud Computing (SoCC)*, pp. 149–161, ACM, 2018.

[4] P. Di Francesco, I. Malavolta, and P. Lago, "Research on architecting microservices: Trends, focus, and potential for industrial adoption," in *Proc. of the IEEE Int. Conf. on Software Architecture (ICSA)*, pp. 21–30, IEEE, 2017.

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

[6] A. de Camargo, I. L. Salvadori, R. dos Santos Mello, and F. Siqueira, "An architecture to automate performance tests on microservices," in *Proc. of the 18th Int. Conf. on Information Integration and Web-based Applications and Services (iiWAS)*, pp. 422–429, ACM, 2016.

[7] P. Bourque and R. E. Fairley, "Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0," *IEEE Computer Society Press*, 2014.

[8] I. Beschastnikh, P. Wang, Y. Brun, and M. D. Ernst, "Debugging distributed systems," *Queue*, vol. 14, no. 2, pp. 91–110, 2016.

[9] N. Alshuqayran, N. Ali, and R. Evans, "A systematic mapping study in microservice architecture," in *Proc. of the 9th Int. Conf. on Service-Oriented Computing and Applications (SOCA)*, pp. 44–51, IEEE, 2016.

[10] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural patterns for microservices: A systematic mapping study," in *Proc. of the 8th Int. Conf. on Cloud Computing and Services Science (CLOSER)*, pp. 1–12, Springer CCIS, 2018.

[11] J. Soldani, D. A. Tamburri, and W. van den Heuvel, "The pains and gains of microservices: A systematic grey literature review," *Journal of Systems and Software*, vol. 146, pp. 215–232, 2018.

[12] F. Paolo Di, L. Patricia, and M. Ivano, "Architecting with microservices: A systematic mapping study," *Journal of Systems and Software*, vol. 150, pp. 77–97, 2019.

[13] G. Márquez, F. Osses, and H. Astudillo, "Review of architectural patterns and tactics for microservices in academic and industrial literature," *IEEE Latin America Transactions*, vol. 16, no. 9, pp. 2321–2327, 2018.

[14] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proc. of the 12th Int. Conf. on Evaluation and Assessment in Software Engineering (EASE)*, pp. 68–77, ACM, 2008.

[15] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Tech. Rep. EBSE Technical Report EBSE-2007-01, Keele University and Durham University, 2007.

[16] V. Braun and V. Clarke, "Using thematic analysis in psychology," *Qualitative Research in Psychology*, vol. 3, no. 2, pp. 77–101, 2006.

[17] B. Beizer, *Software Testing Techniques*. Dreamtech Press, 2003.

[18] M. Palacios, J. García-Fanjul, and J. Tuya, "Testing in service oriented architectures with dynamic binding: A mapping study," *Information and Software Technology*, vol. 53, no. 3, pp. 171–189, 2011.

[19] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proc. of the 18th Int. Conf. on Evaluation and Assessment in Software Engineering (EASE)*, pp. 1–10, ACM, 2014.

[20] B. G. Glaser and A. L. Strauss, *Discovery of Grounded Theory: Strategies for Qualitative Research*. Routledge, 2017.

[21] M. Waseem, P. Liang, G. Márquez, and A. D. Salle, *Testing Microservices Architecture-Based Applications: A Systematic Mapping Study*, 2020. http://doi.org/10.5281/zenodo.3959298.

[22] R. Wieringa, N. Maiden, N. Mead, and C. Rolland, "Requirements engineering paper classification and evaluation criteria: a proposal and a discussion," *Requirements Engineering*, vol. 11, no. 1, pp. 102–107, 2006.

[23] M. Großmann and C. Ioannidis, "Continuous integration of applications for onos," in *Proc. of the 5th Int. Conf. on Network Softwarization (NetSoft)*, pp. 213–217, IEEE, 2019.

[24] J. G. Quenum and S. Aknine, "Towards executable specifications for microservices," in *Proc. of the 14th Int. Conf. on Services Computing (SCC)*, pp. 41–48, IEEE, 2018.

[25] M. Rahman and J. Gao, "A reusable automated acceptance testing architecture for microservices in behavior-driven development," in *Proc. of the 8th IEEE Symp. on Service-Oriented System Engineering (SOSE)*, pp. 321–325, IEEE, 2015.

[26] M. H. Lom, P. M. Ariele, D. R. Fabio, K. Fabrice, H. W. Pierre, F. Riccardo, D. B. Sergio, G. Davide, and L. M., "Automation and intelligent scheduling of distributed system functional testing," *International Journal on Software Tools for Technology Transfer*, vol. 19, no. 3, pp. 281–308, 2017.

[27] M. Peuster, C. Dröge, C. Boos, and H. Karl, "Joint testing and profiling of microservice-based network services using ttcn-3," *ICT Express*, vol. 5, pp. 150–153, 2019.

[28] K. Meinke and P. Nycander, "Learning-based testing of distributed microservice architectures: correctness and fault injection," in *Proc. of the 5th Int. Conf. on Software Engineering and Formal Methods Workshops (SEFMW)*, pp. 3–10, Springer LNCS, 2015.

[29] M. Rahman, Z. Chen, and J. Gao, "A service framework for parallel test execution on a developer's local development workstation," in *Proc. of the 8th IEEE Symp. on Service-Oriented System Engineering (SOSE)*, pp. 153–160, IEEE, 2015.

[30] Y. Jayawardana, R. Fernando, G. Jayawardena, D. Weerasooriya, and I. Perera, "A full stack microservices framework with business modelling," in *Proc. of the 8th Int. Conf. on Advances in ICT for Emerging Regions (ICTer)*, pp. 78–85, IEEE, 2018.

[31] J. Sotomayor, S. C. Allala, P. Alt, J. Phillips, T. M. King, and P. Clarke, "Comparison of runtime testing tools for microservices," in *Proc. of the 43rd IEEE Annual Computer Software and Applications Conf. (COMPSAC)*, pp. 356–361, IEEE, 2019.

[32] T. Takeda, M. Takahashi, T. Yumoto, S. Masuda, T. Matsuodani, and K. Tsuda, "Applying change impact analysis test to migration test case extraction based on idau and graph analysis techniques," in *Proc. of the 12th Int. Conf. on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 131–139, IEEE, 2019.

[33] S.-P. Ma, C.-Y. Fan, Y. Chuang, I.-H. Liu, and C.-W. Lan, "Graph-based and scenario-driven microservice analysis, retrieval, and testing," *Future Generation Computer Systems*, vol. 100, pp. 724–735, 2019.

[34] A. Arcuri, "Evomaster: Evolutionary multi-context automated system test generation," in *Proc. of the 11th Int. Conf. on Software Testing, Verification and Validation (ICST)*, pp. 394–397, IEEE, 2018.

[35] C. Gadea, M. Trifan, D. Ionescu, M. Cordea, and B. Ionescu, "A microservices architecture for collaborative document editing enhanced with face recognition," in *Proc. of the 11th IEEE Int. Symp. on Applied Computational Intelligence and Informatics (SACI)*, pp. 441–446, IEEE, 2016.

[36] R. Pietrantuono, S. Russo, and A. Guerriero, "Run-time reliability estimation of microservice architectures," in *Proc. of the 29th Int. Symp. on Software Reliability Engineering (ISSRE)*, pp. 25–35, IEEE, 2018.

[37] V. Heorhiadi, S. Rajagopalan, H. Jamjoom, M. K. Reiter, and V. Sekar, "Gremlin: Systematic resilience testing of microservices," in *Proc. of the 36th Int. Conf. on Distributed Computing Systems (ICDCS)*, pp. 57–66, IEEE, 2016.

[38] J. Lotz, A. Vogelsang, O. Benderius, and C. Berger, "Microservice architectures for advanced driver assistance systems: A case-study," in *Proc. of the IEEE Int. Conf. on Software Architecture Companion (ICSA-C)*, pp. 45–52, IEEE, 2019.

[39] X. Zhou, X. Peng, T. Xie, J. Sun, C. Ji, W. Li, and D. Ding, "Fault analysis and debugging of microservice systems: Industrial survey, benchmark system, and empirical study," *IEEE Transactions on Software Engineering*, 2018.

[40] D. I. Savchenko, G. I. Radchenko, and O. Taipale, "Microservices validation: Mjolnirr platform case study," in *Proc. of the 38th Int. Conv. on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 235–240, IEEE, 2015.

[41] J. Lehvä, N. Mäkitalo, and T. Mikkonen, "Consumer-driven contract tests for microservices: A case study," in *Proc. of the 19th Int. Conf. on Product-Focused Software Process Improvement (PROFES)*, pp. 497–512, Springer LNCS, 2019.

[42] S. Hacks, A. Steffens, P. Hansen, and N. Rajashekar, "A continuous delivery pipeline for EA model evolution," in *Proc. of the 11th Int. Conf. on Business Process Modeling, Development and Support (BPMDS)*, pp. 141–155, Springer LNBIP, 2019.

[43] L. Chen, "Microservices: architecting for continuous delivery and devops," in *Proc. of the IEEE Int. Conf. on Software Architecture (ICSA)*, pp. 390–397, IEEE, 2018.

[44] O. Zimmermann, "Microservices tenets," *Computer Science-Research and Development*, vol. 32, no. 3-4, pp. 301–310, 2017.

[45] S. Munari, S. Valle, and T. Vardanega, "Microservice-based agile architectures: An opportunity for specialized niche technologies," in *Proc. of the 28th Ada-Europe Int. Conf. on Reliable Software Technologies (Ada-Europe)*, pp. 158–174, Springer LNCS, 2018.

[46] G. Schermann, D. Schöni, P. Leitner, and H. C. Gall, "Bifrost: Supporting continuous deployment with automated enactment of multi-phase live testing strategies," in *Proc. of the 17th International Middleware Conference (MIDDLEWARE)*, pp. 1–14, ACM, 2016.

[47] M. J. Kargar and A. Hanifizade, "Automation of regression test in microservice architecture," in *Proc. of the 4th Int. Conf. on Web Research (ICWR)*, pp. 133–137, IEEE, 2018.

[48] P. Srikaew and I. Kim, "A microservice development for document management system," in *Proc. of the 4th Int. Conf. on Computer Applications and Information Processing Technology (CAIPT)*, pp. 1–4, IEEE, 2017.

[49] Q. Lei., W. Liao, Y. Jiang, M. Yang, and H. Li, "Performance and scalability testing strategy based on kubemark," in *Proc. of the 4th IEEE Int. Conf. on Cloud Computing and Big Data Analysis (ICCCBDA)*, pp. 511–516, IEEE, 2019.

[50] S. Sharma, N. Uniyal, B. Tola, and Y. Jiang, "On monolithic and microservice deployment of network functions," in *Proc. of the 5th IEEE Int. Conf. on Network Softwarization (NetSoft)*, pp. 387–395, IEEE, 2019.

[51] S. P. Ma, C. Y. Fan, Y. Chuang, W. T. Lee, S. J. Lee, and N. L. Hsueh, "Using service dependency graph to analyze and test microservices," in *Proc. of the 42nd IEEE Annual Computer Software and Applications Conference. (COMPSAC)*, pp. 81–86, IEEE, 2018.

[52] M. Camilli, C. Bellettini, L. Capra, and M. Monga, "A formal framework for specifying and verifying microservices based process flows," in *Proc. of the 15th Int. Conf. on Software Engineering and Formal Methods (SEFM)*, pp. 187–202, Springer LNCS, 2017.

[53] H. Schulz, T. Angerstein, D. Okanović, and A. van Hoorn, "Microservice-tailored generation of session-based workload models for representative load testing," in *Proc. of the 27th Int. Symp. on Modeling, Analysis, and Simulation of Computer Telecommunication Systems (MASCOTS)*, pp. 323–335, IEEE, 2019.

[54] J. Campos, Y. Ge, N. Albunian, G. Fraser, M. Eler, and A. Arcuri, "An empirical evaluation of evolutionary algorithms for unit test suite generation," *Information and Software Technology*, vol. 104, pp. 207–235, 2018.

[55] T. Clemson, "Testing strategies in a microservice architecture," 2014. accessed on 2020-03-03.

[56] H. Paul, "Testing challenges related to microservice architecture," 2018. accessed on 2020-02-29.

[57] M. Waseem, P. Liang, and M. Shahin, "A systematic mapping study on microservices architecture in devops," *Journal of Systems and Software*, vol. 170, p. Article No. 110798, 2020.