Testing Microservices Architecture-Based Applications: A Systematic Mapping Study

***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, and challenges in the context of testing MSA-based applications. The search yielded 2,481 articles, of which 31 were finally selected as primary studies. Key findings state 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) inter-communication testing among microservices is gaining the interest of the community. Additionally, it emerges that there is a lack of empirical methods and specific tools to support testing for MSA-based applications. The complexity of MSA-based applications opens new research challenges for introducing more robust and sophisticated interdisciplinary testing strategies.**

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

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].

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 [2]. The number of services, inter-communication processes, dependencies, instances, network communication, and other variables influence testing methodologies in MSA-based applications. 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 importance of microservices research and development in academia and industry alike has led to an increased number of publications describing approaches, challenges, tools, and practices related to this field. Ample evidence of these trends can be found in existing secondary studies (e.g., [3][4][5][6][7]). Nevertheless, 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 31 primary studies, which we examined in detail with the aim of discussing key findings about research themes, testing approaches, and challenges. The main **contribution** of our study is a state-of-the-art concerning 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. [8] for conducting SMSs, together with the strategies presented by Kitchenham et al. [9] 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?* 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. [10] to identify the main research themes related to the primary studies.

RQ2: *What testing approaches have been proposed and used for MSA-based applications?* This RQ aims to collect from the primary studies the techniques, procedures, methods, and other proposals used in the process of testing MSA-based applications.

RQ3: *Which testing-related challenges have been reported in the primary studies over the years?* This RQ aims to illustrate the challenges reported in the primary studies concerning testing of MSA-bases applications. Furthermore, with this RQ, we want to present the evolution of the 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 Appendix of [11]). 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 [6] 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 an 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.

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 labelling 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.

Phase II - Snowballing: To identify more relevant studies, we also applied the snowballing procedure according to the guidelines proposed in [12]. 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 (3 studies), we obtained the final set of included studies (31) for data extraction to answer the RQs.

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 the overview of the selected studies. Concerning data item D8, we applied the approach proposed by Braun et al. [10] to identify main research themes concerning the primary studies. Research themes are related to Thematic Analysis (TA), which is a method for systematically identifying, organizing, and offering insight into patterns of meaning (themes) across a dataset (see Figure 2 in Appendix of [11]). TA is composed of six steps, which are:

1) *Familiarizing with the data*: In this step, the data was read and transcribed regarding data item D8.
2) *Generating initial codes*: The goal of this step was systematically producing the initial list of codes from the transcribed data relevant to research purposes.
3) *Searching for themes*: In this step, generated codes were analyzed and collated into potential themes.
4) *Reviewing themes*: This step checked the suitability of themes in relation to the coded extracts and the entire dataset, aiming to generate a thematic "map" of the analysis.
5) *Defining and naming themes*: During this step, we defined and further refined all the themes under precise and clear names.
6) *Producing the report*: The last step corresponds to introspection. The last analysis was conducted in order to refine themes and characteristics.

Figure 2 in Appendix of [11] describes the process of obtaining research themes executed in this SMS. Two researchers participated in the process in order to reduce bias. The most important activity of this process was brainstorming sessions that were particularly conducted during reviewing, defining, and naming the research themes. In these sessions, both researchers discussed and validated the research themes found. Concerning data item D9 and D10, we executed a sequenced and exhaustive reading on each primary study. For this, we dealt in depth with the structure of the article, identifying ideas that surround testing approaches (D9) and challenges (D10). We identified ideas, secondary ideas, and

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 some of the data items D9 and D10	Studies that do not provide the information regarding the data items from D9 and D10
Study Duration	A study published between 2008 and November 2019	A study published before 2008 or after November 2019

detailed information on each article. Each idea is classified and visualized to establish relationships between the ideas. Due to the similarity of ideas, some of them were consolidated into just one main idea.

Finally, we created a replication package [11] which contains the detailed protocol of our SMS with all the significant data (e.g., PDF files, template, and others) related to the SMS.

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. [10]	RQ1
D9	Testing approaches	What testing strategies (unit testing, integration testing, consumer testing) are used to evaluate the MSA based system?	RQ2
D10	Challenges	What challenges are reported in primary studies?	RQ3

III. RESULTS

In this section, we proceed to answer the research questions according to the data collected in Table III.

A. *Demographics*

We obtained a total of 31 primary studies published from 2015 to November 2019 (see Figure 1). It is clear to see a significant increase in the number of publications related to testing of MSA-based applications from 2017 to 2019. This publication's increment occurs because specialized conferences (e.g., SOSE, SCC, COMPSAC) increase the number of publications about microservices research since the year 2016.

On the other hand, the result shows (see Figure 3 in Appendix of [11]) that most (24 out of 31, 77.4%) of primary studies were published in conferences followed by journals (4

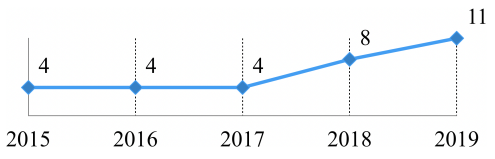

Fig. 1. Studies distribution over publication years

out of 31, 12.9%) and workshops (3 out of 31, 9.6%). The venues result 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. [13] for articles classification (see Figure 2). Figure 2 shows that over half of the primary studies are related to 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 investigated the properties of a solution proposal that has not yet been implemented (Validation research), (ii) some studies contain the authors' opinions about what is wrong or right about something (Opinion paper), and (iii) some other studies discuss the investigation of a problem or an implementation of a technique in practice (Evaluation research).

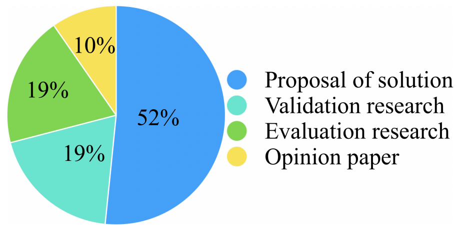

Fig. 2. Research type classification of the primary studies

B. *RQ1: Research Themes*

We identified five main themes that characterize the primary studies such as performance, architecture, DevOps and CI, automated testing, and model-based testing. Subsequently, Table IV describes the sub-themes and their corresponding main points. Research themes represent the main pillars that support primary studies. In turn, each theme is divided into sub-themes that represent key research topics, related to testing, that each primary study addresses. In the following, we further describe each research theme.

*a) **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. Moreover, the question that the primary studies illustrate, in general, is how to partition a system with MSA in order to create test cases and thus be able to test them. Some primary studies such as [14] and [15], discuss this topic.

*b) **DevOps and CI**:* DevOps consists of a set of best practices (e.g., Continuous Integration (CI), Continuous Delivery (CD), testing, deployment) intended to develop, test, and deploy software changes quickly and reliably by promoting strong collaboration between development and operational personnel. This theme includes the studies that report the testing approaches and tools used for MSA-based applications in DevOps or CI context [16].

*c) **Automated testing**:* This theme covers the primary studies which discuss specific tests via automated testing. As an example, Quenum and Aknine presented an empirical approach (i.e., LASTA automated testing) based on intelligent agents for microservice automated testing. These intelligent agents are introduced as abstractions for the services. By using this approach, developers derive the unit and acceptance tests from the formal specification for microservices [17]. Similarly, Shang et al. also introduced a graph-based and scenario-driven scheme to analyze, test, and reuse microservices. This approach enables the automatic retrieval of test cases required to deal with microservice changes [18].

*d) **Performance**:* This theme deals with the quantitative behavioral aspects of testing. The result shows that the primary studies related to this topic have the focus of attention to quantitative performance testing, mostly in the production phase of MSA-based applications development. For instance, Camargo et al. presented a novel approach for automatic execution of performance tests to evaluate the performance of individual microservice by integrating test specifications [19]; Jayawardana et al. proposed a microservices framework that (also) helps to evaluate the scaling capabilities of microservices by using unit testing [20].

*e) **Model-based testing (MBT)**:* This theme gathers those primary studies that discuss the MBT in the context of microservices. For example, Camilli et al. [21] present a formal semantics based on Petri nets models that are applicable in the context of microservices testing. There is one other study that uses a model-based approach to generate workload models as representative of load testing. These models are session-based and represent the workload of one or more specified microservice(s) [22].

Key finding of RQ1: Primary studies reveal novel testing approaches in CD/CI practices.

C. RQ2: Testing Approaches

The main testing approach mentioned in primary studies is Integration Testing (51.6%, 16 out of 31). The main goal of this type of test is to verify the correct assembly between the different components. In particular, it verifies the correct 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. In this regard, the following scenarios describe integration tests in MSA-based applications in the considered primary studies:

- To check how each service serves with other services and other external components.
- To check new features added to the MSA-based application.
- To test service-to-service communication.

Subsequently, the testing approach, which is also highly mentioned in primary studies, is Unit Testing (38.7%, 12 out of 31). 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).

The rest of the testing approaches illustrated in Figure 3 indicate that primary studies have discussed various types of testing. This is because, mainly, running testing in MSA-based applications is challenging due to the complexity of the coordination and deployment of services. Hence, the results of this RQ indicate that the Integration and Unit Testing are showing positive results when faced with the problem concerning the complexity of the services and deployment; however, they are not enough to thoroughly test an MSA-based application.

Key finding of RQ2: The complexity of executing test cases in MSA-based applications lies in the coordination of services and their deployment. In this regard, Integration and Unit Testing turn out to be more favorable to address this circumstance.

D. RQ3: Challenges Related to Testing of MSA-based Applications

The primary studies reported the following challenges related to testing MSA-based applications:

- *Acceptance testing*: Acceptance tests establish the degree of confidence in a system, parts of it, or its non-functional characteristics. The trust in the system will be determined by its degree of adherence to the requirements and business processes requested by the user or client.
- *Automated testing*: Techniques, based on a certain degree of automation, to test software. It automates repetitive tasks and other testing tasks, which are difficult to perform manually.
- *Faster test feedback*: Manual or automatic testing procedures whose purpose is to provide feedback concerning the performance of an application in early development phases. Existing testing framework does not provide reliable faster test feedback for systems based on file system access,

TABLE IV
A CLASSIFICATION OF RESEARCH THEMES ON TESTING MSA-BASED APPLICATIONS FROM THE PRIMARY STUDIES

Themes	Sub-themes	Key points	Study
Architecture	System level test cases	Creation of system level test cases using evolutionary algorithms	[14]
	Test containers	Testing of individual services as separate containers enabling seamless deployment	[15]
	Scalability	Unit testing to evaluate each service	[20]
	Reliability	Testing method for on-demand reliability estimation of microservice applications	[23]
	Behavior-Driven Development	Acceptance tests that describe the expected behavior of a feature	[24]
	Resilience	Testing the resiliency of service in production infrastructures	[25]
	Black box testing	Black box testing for each service	[26]
	Testing approaches	Service and end user/client application testing	[27]
	Learned-based testing	Evaluation of the functional correctness of distributed systems to injected faults	[28]
	Cloud applications testing	Develop the microservice validation methodology for microservice testing	[29]
	MSA evaluation	Design, transform, implement, and test an advanced driver assistance system (ADAS) based on MSA	[30]
	Consumer-Driven contract testing	Testing of MSA-based applications can be improved with contracts	[31]
DevOps and CI	Continuous delivery	Development of regression test for microservices	[32]
	Continuous deployment	Deployment testing in production environments	[33]
	Continuous integration	Development, integration, and testing of Applications for ONOS	[16]
	Continuous delivery	A continue delivery pipeline for design, development, and testing of MSA	[34]
	Integrated software development	Tests in execution in a cluster environment	[35]
Automated testing	Intelligent agents	Encapsulate the formal specifications of services	[17]
	Functional test automation	Test automation of services (black-box testing) and service architectures (grey-box testing)	[36]
	Functional test	Usage of TTCN-3, a testing language defined by the European standards institute (ETSI)	[37]
	Automation framework	Automatically verify software properties	[38]
	Classification of tools	comparison of open-source tools used to support the testing of microservices	[39]
	Migration test case extraction	Usage of Impact data All Used (IDAU) and Graph Analysis Techniques	[40]
	Service Dependency Graph (SDG)	Graph-based and Scenario-driven Microservice Analysis, Retrieval and Testing	[18]
Performance	Performance tests	Evaluation of the performance that each service can deliver	[19]
	Performance testing comparison	Performance comparison of monolith/microservice in Network Function Virtualization (NFV)	[41]
	Parallel testing	Ensuring isolation of processes for each test group running concurrently	[42]
	Performance analysis	Method of performance testing with Kubemark	[43]
Model based testing	Dependency graph	Graph-based microservice analysis and testing approach	[44]
	Petri nets	Petri nets as the basis of model-based testing	[21]
	Log and Model based algorithms	Microservice-tailored generation of Session-based Workload models for representative load testing	[22]

database fixtures and network communication (e.g., MSA based applications) [42].

- *Granularity testing*: Testing methods and techniques which focus on testing the most specific functionalities of the software.Granularity testing is challenging because of chaining interfaces and asynchronicity complexities of MSA based applications [15].
- *Integration testing*: The integration testing challenge could occur because of combinatorial expansion between the microservices interactions. 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 [45].
- *Inter-communication testing*: Testing methods and techniques whose purpose is to test communication mechanisms (synchronous or asynchronous) that involve inter intercommunication between services.
- *Monitoring testing*: Testing approaches whose purpose is to test the tracking process an application.
- *Performance testing*: These tests aim to optimize the software from the point of view of the effectiveness of the applications.
- *Runtime testing*: Testing approaches aimed at finding errors, defects, and failures on software at runtime.

Figure 4 in Appendix of [11] illustrates the challenges mentioned in the primary studies over the years.

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 process automation techniques in order to reduce human intervention on software testing. In the context of microservices, automated testing becomes a great ally to test MSA-based applications.

The communication between microservices can be synchronous (the client sends a request and waits for a response from the service) or asynchronous (the client code or message sender usually does not wait for a response). One of the main features of MSA-based applications is that they use light-weight communication mechanisms. Therefore, there is an emerging interest in test communication protocols (such as AMQP, HTTP / HTTPS, TCP) among microservices.

Key finding of RQ3: There is a growing interest in testing types of communication (synchronous and asynchronous) between microservices.

IV. DISCUSSION

A. Lack of Empirical and Experimental Studies

Figure 4 summarizes the empirical strategies used by the primary studies to validate their respective proposals in the context of testing MSA-based applications. According to the figure, case studies predominate as the most frequently used empirical strategy. In these studies, testing approaches are used as a variable to evaluate proposals that are not directly related to testing (for example, business modeling

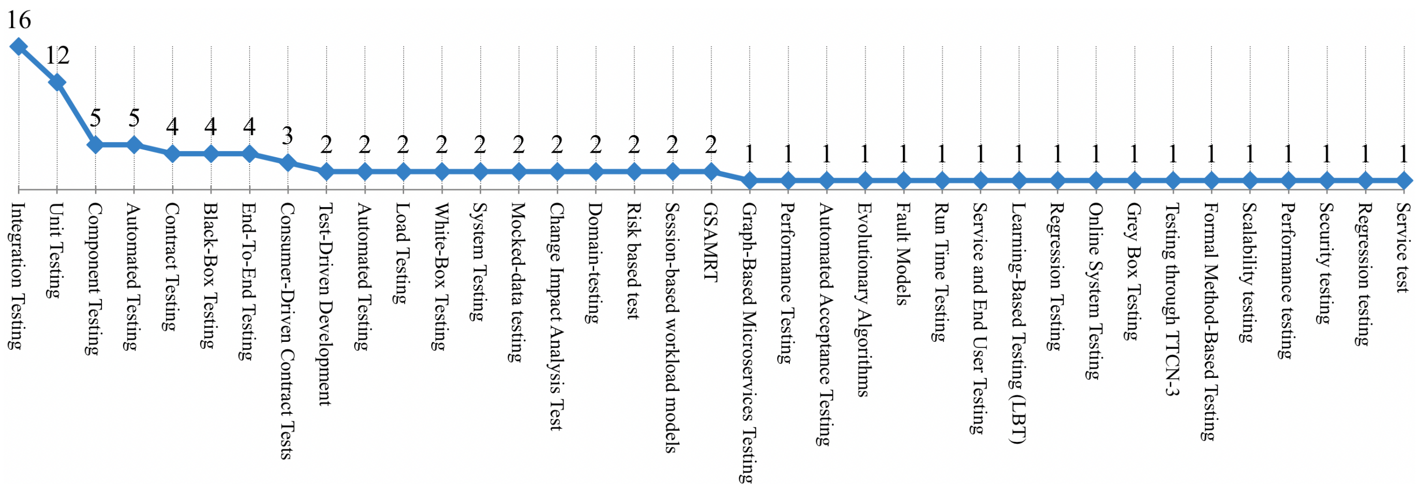

Fig. 3. Testing approaches for MSA-based applications from the primary studies

[20], face recognition [15], process flow [21], among others). Nevertheless, 26% of studies have not mentioned the empirical strategies used to evaluate proposals directly related to testing of MSA-based applications.

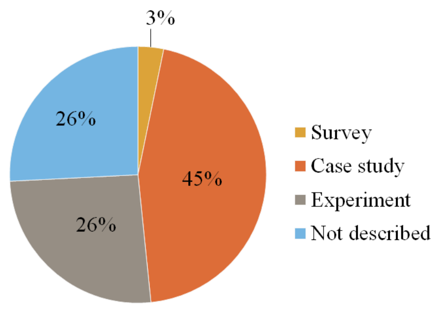

Fig. 4. Empirical strategies reported in the primary studies

Pietrantuono et al. [23] developed an in-vivo testing algorithm based on an adaptive web sampling strategy, named Microservice Adaptive Reliability Testing (MART). This algorithm is formulated as an adaptive sampling scheme that navigates the input space of the application to draw test cases with a higher chance of improving the reliability estimate.

Ma et al. [44] proposed GMAT, a graph-based technique for testing in MSA-based applications. GMAT aims to assist the development of MSA-based applications by generating services dependency graphs and visualizing the dependency relationships between microservices. The goal of experimentation in this study is to evaluate the performance of GMAT based on simulations. The reason why authors created simulations to evaluate the GMAT performance is that they could not obtain the source code and configurations of real-world MSA-based applications.

Meinke et al. [28] evaluated the functional correctness of distributed systems and their robustness to injected faults by applying Learning-Based Testing (LBT). The authors described an experiment that performed requirements testing and fault injection on a distributed MSA for counter-party credit risk analysis known as triCalculate. Other empirical strategies (such as demonstrations) mostly focus on (i) evaluating service frameworks for parallel test execution in a developer's containerized sandbox using operating system-level virtualization provided by Docker [25] and (ii) interviews and surveys with practitioners [27].

Although the primary studies that employ case study approach (45% of the primary studies) provide a series of theories about testing approaches for MSA-based applications, there is a deficit in empirical and experimental studies that allow developing analytical techniques and methods regarding testing approaches for MSA-based applications. Also, the primary studies use well-known testing approaches for MSA-based applications, but there is no certainty that these approaches are necessary enough to support the development of MSA-based applications.

B. *Lack of Tools for Testing MSA-based Applications*

Table V summarizes the tools that are mentioned in the primary studies. We classified the tools based on the testing strategies collected by Clemson [46] for MSA-based applications, which are unit testing, integration testing, component testing, contract testing, and end-to-end testing. We observe that most of the identified tools are not particularly developed to test MSA based applications, but these tools support the testing of MSA based systems. On the other hand, we did not find primary studies whose testing approach is based on testing portions of MSA-based applications.

Although Table V shows 5 testing tools that can help to evaluate different aspects of MSA-based applications, more tools should be developed to perform robust tests. Furthermore, MSA-based applications have other systemic properties (such as scalability, availability, and others), which are necessary and critical to evaluate.

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

V. Threats to Validity

Threats to *internal validity* describe factors that could affect the results obtained from the study. We addressed the following threats with specific measures:

- Bias on study selection: The selection of studies has been made by applying explicitly defined inclusion and exclusion criteria (see Table II). To avoid the possible bias, we also performed the cross-check validation for all selected studies.
- Bias on data extraction: To obtain data consistency and avoid bias on data extraction, we defined the data extraction form (see Table III). Initially, two authors equally distributed the number of studies and then extracted the data according to the data extraction form. The two authors regularly discussed and shared their findings to avoid data extraction bias.
- Bias on study themes classification: We identified the research themes by using the guidelines of thematic analysis proposed by Braun et al. [10]. Furthermore, these guidelines provide qualitative analytic methods to obtain research themes in the primary studies.

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 [11] that rigorously specifies the whole process of conducting this SMS.

Threats to the *conclusion validity* are concerned with issues that affect the ability to draw the correct conclusions. Although we followed the guidelines for performing SLRs by Kitchenham et al. [9], which already assumes that not all relevant primary studies that exist can be identified, we handled this validity threat by discussing our results in several brainstorming sessions. The number of primary studies obtained in this SMS allowed us to analyze each primary study critically.

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 the exclusion of relevant papers and the inclusion of 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 database format, and (iii) executing the customized search string on popular databases.

VI. Related Work

Alshuqayran et al. [3] 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. [4] and Márquez et al. [7] reported architectural patterns and tactics related to microservices.

The study conducted by Soldani et al. [5] reports pains and gains from industrial grey literature related to the design, development, and operation of MSA-based applications. Similarly, Di Francesco et al. [6] present a comprehensive review of architecting microservices concerning the focus of research, industrial adoption, and publication trends.

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 article describes an SMS regarding testing approaches used in MSA-based applications. From an initial set of 2,481 retrieved articles, we obtained 31 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 Performance, Architecture, DevOps/CI, Automated Testing, and Model-based Testing. On the other hand, key findings obtained from the results point to (i) it is necessary to perform more empirical and experimental studies in order to validate proposals related to testing in MSA-based applications and (ii) more testing tools should be created to evaluate and analyze systemic properties (e.g., scalability, availability) of MSA-based applications.

Finally, our future work focuses on knowing the industrial reality concerning testing for MSA-based applications. We intend to conduct surveys to understand the practitioners' point of view regarding testing in microservices and to what extent the research outcomes have been employed in practice.

References

[1] S. Newman, "Building microservices: designing fine-grained systems.," *O'Reilly Media, Inc.*, 2015.

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

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

[4] D. Taibi, V. Lenarduzzi, and C. Pahl, "Architectural patterns for microservices: A systematic mapping study," *Proceedings of the 8th International Conference on Cloud Computing and Services Science (CLOSER)*, pp. 1–12, 2018.

[5] J. Soldani, D. A. Tamburri, and W. J. Van Den Heuvel, "The pains and gains of microservices: A systematic grey literature review," *Journal of Systems and Software*, vol. 146, no. 215-232, 2018.

[6] 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.

[7] G. Márquez, F. Osses, and H. Astudillo, "Review of architectural patterns and tactics for microservices in academic and industrial literature," *XXI Ibero-American Conference on Software Engineering (CIbSE)*, pp. 71–84, 2018.

[8] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," *Proceedings of the 12th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, vol. 8, no. 68-77, 2008.

[9] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," *Keele University and Durham University Joint ReportTech.rep.ebse 2007-001*, 2007.

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

[11] *Testing Microservices Architecture-Based Applications: A Systematic Mapping Study: Replication Package*. https://tinyurl.com/rxvy7uu.

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

[13] 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.

[14] A. Arcuri, "Evomaster: Evolutionary multi-context automated system test generation," *Proceedings of the 11th International Conference on Software Testing, Verification and Validation (ICST)*, pp. 394–397, 2018.

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

[16] M. Großmann and C. Ioannidis, "Continuous integration of applications for onos," pp. 213–217, 2019.

[17] J. G. Quenum and S. Aknine, "Towards executable specifications for microservices," *Proceedings of the 14th International Conference on Services Computing (SCC)*, pp. 41–48, 2018.

[18] 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.

[19] A. de Camargo, I. Salvadori, R. D. S. Mello, and F. Siqueira, "An architecture to automate performance tests on microservices," *Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services (iiWAS)*, pp. 422–429, 2016.

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

[21] M. Camilli, C. Bellettini, L. Capra, and M. Monga, "A formal framework for specifying and verifying microservices based process flows," *Proceedings of the 15th International Conference on Software Engineering and Formal Methods (SEFM)*, pp. 187–202, 2017.

[22] H. Schulz, T. Angerstein, D. Okanović, and A. van Hoorn, "Microservice-tailored generation of session-based workload models for representative load testing," pp. 323–335, 2019.

[23] R. Pietrantuono, S. Russo, and A. Guerriero, "Run-time reliability estimation of microservice architectures," *Proceedings of the 29th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 25–35, 2018.

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

[25] V. Heorhiadi, S. Rajagopalan, H. Jamjoom, M. K. Reiter, and V. Sekar, "Gremlin: Systematic resilience testing of microservices," *Proceedings of the 36th International Conference on Distributed Computing Systems (ICDCS)*, pp. 57–66, 2016.

[26] P. Srikaew and I. Kim, "A microservice development for document management system," *Proceedings of the 4th International Conference on Computer Applications and Information Processing Technology (CAIPT)*, pp. 1–4, 2017.

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

[28] K. Meinke and P. Nycander, "Learning-based testing of distributed microservice architectures: Correctness and fault injection," *SEFM Collocated Workshops*, pp. 3–10, 2015.

[29] D. I. Savchenko, G. I. Radchenko, and O. Taipale, "Microservices validation: Mjolnirr platform case study," *Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 235–240, 2015.

[30] J. Lotz, A. Vogelsang, O. Benderius, and C. Berger, "Microservice architectures for advanced driver assistance systems: A case-study," pp. 45–52, 2019.

[31] J. Lehvä, N. Mäkitalo, and T. Mikkonen, "Consumer-driven contract tests for microservices: A case study," pp. 497–512, 2019.

[32] M. J. Kargar and A. Hanifizade, "Automation of regression test in microservice architecture," *Proceedings of the 4th International Conference on Web Research (ICWR)*, pp. 133–137, 2018.

[33] L. Chen, "Microservices: architecting for continuous delivery and devops," *Proceedings of the 2nd International Conference on Software Architecture (ICSA)*, pp. 39–397, 2018.

[34] S. Hacks, A. Steffens, P. Hansen, and N. Rajashekar, "A continuous delivery pipeline for ea model evolution," pp. 141–155, 2019.

[35] D. Liu, H. Zhu, C. Xu, I. Bayley, D. Lightfoot, M. Green, and P. Marshall, "Cide: An integrated development environment for microservices," *Proceedings of the 13th International Conference on Services Computing (SCC)*, pp. 808–812, 2016.

[36] 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.

[37] 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, 2019.

[38] S. Munari, S. Valle, and T. Vardanega, "Microservice-based agile architectures: An opportunity for specialized niche technologies," *Ada-Europe International Conference on Reliable Software Technologies (Ada-Europe)*, pp. 158–174, 2018.

[39] J. Sotomayor, S. C. Allala, P. Alt, J. Phillips, T. M. King, and P. Clarke, "Comparison of runtime testing tools for microservices," vol. 2, pp. 356–361, 2019.

[40] 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," pp. 131–139, 2019.

[41] S. Sharma, N. Uniyal, B. Tola, and Y. Jiang, "On monolithic and microservice deployment of network functions," pp. 387–395, 2019.

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

[43] Q. Lei., W. Liao, Y. Jiang, M. Yang, and H. Li, "Performance and scalability testing strategy based on kubemark," pp. 511–516, 2019.

[44] 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," *Proceedings of the 42nd Annual Computer Software and Applications Conference (COMPSAC)*, pp. 81–86, 2018.

[45] H. Paul, "Testing challenges related to microservice architecture," 2018. Accessed = 2020-02-29.

[46] T. Clemson, *Testing Strategies in a Microservice Architecture*, 2014. https://martinfowler.com/articles/microservice-testing/.