Research Opportunities in Microservices Quality Assessment: A Systematic Literature Review

Verónica C. Tapia [1,2,*] and Carlos M. Gaona [2]

[1] Sistemas de Información/Universidad Técnica de Cotopaxi, Latacunga, Ecuador
[2] Escuela de Ingeniería de Sistemas y Computación/Universidad del Valle, Santiago de Cali, Colombia;
Email: mauricio.gaona@correounivalle.edu.co (C.M.G.)
*Correspondence: veronica.tapia@utc.edu.ec (V.C.T.)

***Abstract*—The growth in the development of microservices has sparked interest in evaluating their quality. This study seeks to determine the key criteria and challenges in evaluating microservices to drive research and optimize processes. The systematic review of the literature presented in this research identified that the most commonly used evaluation criteria are performance, scalability, security, cohesion, coupling, and granularity. Although evaluation tools exist, they mainly measure performance aspects such as latency and resource consumption. Challenges were identified in security, granularity, throughput, monitoring, organizational strategy, orchestration, choreography, scalability, decomposition, and monolith refactoring. In addition, research opportunities in empirical studies, analysis of quality trade-offs, and broadening of relevant perspectives and tools are noted. Challenges in the interrelation of quality attributes, metrics and patterns, automatic evaluation, architectural decisions and technical debt, domain-based design, testing, monitoring, and performance modeling are also highlighted. Challenges in orchestration, communication management and consistency between microservices, independent evolution, and scalability are also mentioned. Therefore, it is critical to address these particular challenges in microservices and to continue research to improve the understanding and practices related to quality.**

***Keywords*—attributes, challenges, evaluation tools, quality metrics, software**

I. Introduction

Microservice Microservices (MS) development is based on an architectural and organizational approach in which applications are composed of small independent services that can be built on different platforms and through multiple technological tools. Each service executes a unique functionality in its process. Microservices are an interesting option for those who wish to migrate their systems to improve their application performance, maintainability, scalability, and interoperability.

As with all types of software, it is essential to incorporate the aspects of MS development to ensure quality. However, this can be much more complex in MS. Hence, it is a crucial concern for researchers. Although contributions have been reported, even with tools that facilitate migration and development processes while preserving quality attributes, there is still no clarity in this field [1].

The objective of this work is to contribute to the understanding of the processes related to the quality of microservices. For this purpose, a scientific literature review on the most important attributes and metrics was performed, and research challenges in this area were identified. Moreover, this study is directed at the community of developers of microservice-based systems, academia, and the software industry in general.

A Systematic Literature Review (LRS) was conducted to answer five questions to identify the attributes, metrics, challenges, and tools used to assess MS quality.

The Research Questions (RQ) posed are:

RQ1: What are the quality attributes applicable to microservices?

RQ2: What are the quality metrics applicable to microservices?

RQ3: What challenges have been identified in microservices quality assessment?

RQ4: What are the most used microservices assessment tools?

RQ5: What metrics do the tools use?

This paper is organized as follows: Section I comprises a concise literature review, the objective of the study, and the research questions; Section II describes the basic concepts of microservices, software quality, quality applicable to microservices, and specific works related to the study of microservices quality attributes and metrics are discussed; Section III describes the methodological process used in this study; Section IV presents the results of the study, the general results are discussed first, then each research question results citing the most important references; Finally in Section V, the conclusions are presented.

II. Literature Review

A. Microservices-Based Applications

MS-based applications are an architectural and organizational approach to software development in which applications consist of small independent services that

Manuscript received February 27, 2023; revised April 20, 2023; accepted May 29, 2023; published September 25, 2023.

communicate through RESTful or RPC-based Application Programming Interface (APIs) and lightweight protocols [2, 3].

It is a collection of self-contained services that work together to provide functionality. Currently, there are many options for building microservices, some of which are pattern driven. However, the mapping between attributes and quality patterns remains unclear [4].

Microservice architecture is a software development technique for creating interrelated applications that operate independently and ideally have low coupling (independence between microservices, such that they can operate autonomously without altering each other's functionality) and provide lightweight services. This method of conceiving applications is ideally suited to distributed architectures in the cloud because each service is implemented in different execution environments belonging to different physical infrastructures. The rationale of the microservice concept is to perform small tasks to reduce complexity. In this architecture, several services interact with each other. Among the main characteristics of microservices are: maintainability, evolvability, independent deployment, low costs as a single service, improvements, and shorter time to market [5].

B. Software Quality

Software quality determines the performance of a product's features and its ability to satisfy the expressed and tacit needs of the users. An effective development process allows us to obtain a useful product, providing measurable value for those who produce and manage it [6].

The ISO/IEC 25000 family of standards, known as SQuaRE (System and Software Quality Requirements and Evaluation), are used to measure quality. SQuaRE is a set of rules that aim to create a common framework for assessing a software product's quality, which results from the evolution of previous standards, especially ISO/IEC 9126 and ISO/IEC 14598 [7].

Quality is defined by quality attributes (performance, testability, security, and availability) using metrics that evaluate persistence.

C. Quality Attributes

These are the general factors that affect the runtime behavior of a system, its design, and user experience. They describe the purpose of the system in the environment in which it is built. From a technical perspective, quality attributes drive important architectural and design decisions. They are classified into two types: in the first type, the quality attribute maps to several specific requirements; in the second type, the quality attribute does not map to specific functional requirements, but it affects all functional requirements [8, 9].

ISO 25000 [7] refers to the following quality attributes: functionality, performance efficiency, compatibility, usability, reliability, security, maintainability, and portability.

Quality attributes represent the properties of a system that do not determine the functionality. Otherwise, they define the quality and characteristics that the system must support. They are generally divided into sub-attributes to facilitate measurement. For example, accessibility is a sub-attribute of usability that identifies the level of satisfaction of stakeholders' needs and the degree of success of a system [2, 3].

In agile development projects, the definition of requirements mainly focuses on functional characteristics. Thus, the manifestation of quality attributes and the traceability of the relationships between requirements are challenging. It hinders the maintenance of the system because the significance of the effects of changes in functional requirements is unknown. Therefore, it becomes a priority that agile methodologies, such as Scrum, incorporate the analysis of quality attributes in the development plans. As a result, when projects fail, it is usually not due to their defective functionalities but to problems in their quality attributes, such as complicated maintenance, low performance, and security conflicts [1].

Measurement in software engineering is crucial for evaluating quality characteristics and setting goals. Measurement is defined as a logical sequence to quantify properties on a specific scale, and measurement functions or algorithms can combine quality measurement elements [4]. However, some critical design methods have been omitted from the architecture measurement process.

The main objectives of software quality measurement include assisting management in monitoring and controlling development and maintenance, observing conformance to requirements, and serving as a data source for improvement. Some quality attributes can be measured using architectural characteristics such as size, complexity, coupling, and cohesion. These measurements are defined by standards such as IEC 25023.

D. Quality Attributes in Microservices.

According to Valdivia *et al.* [10], who studied architecture patterns, the quality attributes of MS identified in the literature are maintainability, reliability, security, performance, compatibility, and portability. These are measurable non-functional requirements whose purpose is to determine the level of user satisfaction [11]. Similarly, Richardson [12] stated that microservices must be maintainable, testable, coupled, independently implementable, and capable of being developed by small teams. Finally, Osses *et al.* [13] described exploratory research based on a systematic literature review that yielded 44 architectural patterns and five quality attributes (scalability, flexibility, testability, performance, and elasticity), which are related to most of the architectural patterns and tactics for microservices.

E. Quality Metrics

Metrics evaluate the degree of presence of quality attributes, which can be internal when they do not depend on the software execution (static) or external when they apply to the running software (dynamic), allowing quantification of the attributes by providing a numerical value as a measure [9, 10].

Selmadji and Seriai *et al.* [14] determined a set of quality evaluation criteria classified according to static and dynamic analysis presented in Table I.

TABLE I. QUALITY ASSESSMENT METRICS BY TYPE OF ANALYSIS [5]

Static Analysis	Dynamic Analysis
Granularity	Execution cost
LOC	Response Time
Open interfaces	Availability
High cohesion	Succ. Exec. Rate
Loose coupling	Usage Frequency
	Scalability
	Independence
	Maintainability
	Deployment
	Health Management
	Modularity
	Manageability
	Performance
	Reusability
	Tech. Heterogeneity
	Agility
	Security
	Load Balancing
	Org. Alignment

F. Related Work

Several studies have been conducted on microservice quality involving metrics and attributes, some of which are discussed below.

Jin and Liu *et al.* [15] discusses the popularity of microservices in software architecture and the challenges they present, such as resilience, performance, and evolution. It describes a systematic mapping that analyzes how self-adaptive approaches are applied in microservices-based architectures. This paper selected twenty-one studies between February and May 2020 through an automated search of five scientific databases. It also proposes four promising research directions. The main findings indicate that most studies focus on self-healing, performance quality, and scalability attributes.

Li *et al.* [1] systematically analyzes quality in Microservice-based Architectures (MSAs). It identified six principal quality assurances and 19 tactics that architecturally address these assurances. The study contributes to the knowledge of the state of the art of quality assurances in MSAs. It may be helpful for software practitioners and architects in decision-making. In addition, it identifies missing areas and opportunities for research on the topic. The authors plan to conduct primary empirical studies and expand the review, focusing on the latest knowledge additions.

Valdivia and Lora-González *et al.* [10] was oriented toward answering three questions linked to the patterns used in microservices, quality attributes, and metrics to quantify the quality attributes related to a pattern. The work includes gray literature, and as a result, 54 patterns aimed at solving communication difficulties and optimizing performance were identified; reliability and security as the quality attributes most frequently related to patterns. As for metrics, time, percentage of accepted requests, number of files to be modified, and energy consumption were identified. The publication aims to improve the understanding of microservices-based system design through a systematic, multivocal literature review of patterns related to microservices architecture and their correspondence to quality attributes and metrics identified in academic and industrial research.

Gorski and Wozniak [16] notes that numerous articles have been written on optimizing business process execution in service architecture with reliability in mind. In this study, publications from 2006 to 2020 were reviewed. Therefore, 128 relevant articles were selected, providing a broader view of the field of application of services to support business processes. The reviewed articles describe different methods related to optimization steps, such as resource allocation, service composition, and service scheduling. Additionally, 119 out of 128 articles focus on service composition methods, while only three address service schedule, indicating that this stage is the least developed in optimization. The most popular method identified in the articles is the genetic algorithm, which many researchers consider highly effective. In addition, the review includes articles describing new original heuristic algorithms that are relevant to solving problems at the service composition stage.

Capuano and Muccini's [17] objective was to determine the impact of a quality-driven approach in migrating to microservices. An LRS was proposed with three questions: studies implementing a quality-driven approach to migrate to microservices, quality attributes analyzed in the migration phases, and the migration phase in which the quality-driven process is implemented. The results revealed that quality-driven migration is growing; 28.21% of the researchers identified quality attributes in the understanding phase, 48.72% in the microservice identification phase, 35.90% in the microservice evaluation phase, and 2.56% in the packaging phase. This finding emphasizes that many researchers have considered quality attributes in their studies. However, it appears that quality improvement is not the migration objective.

Li *et al.* [1] aimed to investigate the state-of-the-art microservice quality attributes, which allowed the identification of studies on testing related to MS quality assurance. The results provide an overview of the six quality attributes of most concern in MS: scalability, performance, availability, monitoring, security, and testability; they identify 19 tactics that architecturally address critical quality assurances, including two tactics for scalability, four for performance, four for availability, four for monitoring, three for security, and two for testability.

III. MATERIALS AND METHODS

The LRS process began in July 2022 based on an adaptation of the methodology proposed by Kitchenham [18]. Nine stages were executed, Fig. 1, considering the five research questions. From Stage 4 onwards, the process was carried out iteratively for each question see Fig. 2.

A. Research Questions

Considering that the objective of this study was to understand and interpret the processes of MS quality, five research questions were posed to conduct an LRS, Table II.

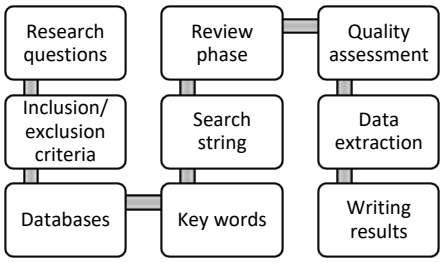

Figure 1. Stages of the systematic literature review.

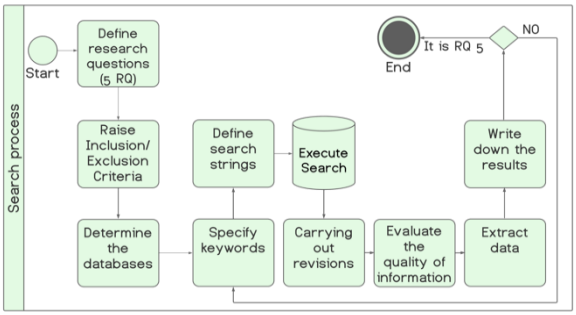

Figure 2. Information search process.

TABLE II. RESEARCH QUESTIONS

Questions	Motivation
RQ1: What are the quality attributes applicable to microservices?	The objective of the first question is to know the quality attributes most investigated in MS and those of concern to the developer community.
RQ2: What are the quality metrics applicable to microservices?	The second question requires identifying the parameters applied to evaluate MS quality, i.e., the metrics used
RQ3: What challenges have been identified in assessing the quality of microservices?	The third question was posed because of the need to know the gaps in the research on MS quality assessment to project future work and raise them to the research community.
RQ4: What are the most used microservices evaluation tools?	The fourth question aims to identify tools to evaluate the quality of microservices.
RQ5: What metrics do the tools use?	Once the tools have been identified, the fifth question's objective is to know the metrics used to evaluate MS quality.

B. Document Search Process

In this phase, the inclusion and exclusion criteria were established to consider valid sources: publications from 2017 onwards, number of citations, and full versions of the articles. Gray literature (technical reports, abstracts, or presentations) and articles not related to software and microservices were excluded, Table III.

C. Databases

Databases with the highest bibliographic production in microservices and quality attributes were delimited and included in the selection: Scopus, ScienceDirect, IEEE Xplore Digital Library, Web of Science, Springer, and ACM Digital Library.

D. Keywords

During this phase, the search terms in each database were specified. The keywords used in this study are microservices, quality, microservice quality, quality attributes, quality metrics, microservice quality challenges, and microservice quality assessment tools.

E. Search Strings

A list of search strings was defined according to each question, and the relationships between the keywords were specified using AND, OR, and NOT logical operators. For example, for research question 1, what are the quality attributes applicable to microservices? The list of keywords contains microservices and quality attributes, the search string in Scopus, considering publications since 2017, is: TITLE-ABS-KEY (((("Quality Attribute") AND (("Microservices")) OR ("Microservice quality attributes")) AND PUBYEAR > 2016 AND PUBYEAR < 2024. Similarly, in ScienceDirect, ("Quality Attribute" AND "Microservices") OR ("Quality attributes in Microservices" OR "Quality of Microservices") Year: 2017-2023.

F. Review Phase

The databases were reviewed using the different search strings and inclusion/exclusion criteria.

G. Quality Assessment

The quality of the information collected was assessed by verifying the relevance of the content to answer each question, the study's objectives, methodology, and results.

H. Data Extraction

All the information that passed the quality assessment was recorded in the database created for this purpose. The pertinent information was extracted from the selected sources using detailed research questions and objectives analysis.

I. Writing Results

Finally, the article is written, and the results are presented as graphs, tables, and detailed explanations.

TABLE III. INCLUSION AND EXCLUSION CRITERIA

Inclusion Criteria	Exclusion Criteria
IC1: Articles published in recent years, as of 2017. IC2: Published articles with the highest number of citations. IC3: If an article describes more than one study, each study is evaluated individually. IC4: If there are short and full versions of the same study, the latter is included	EC1: Technical reports and papers available as abstracts or presentations (gray literature). EC2: Articles that do not present studies related to software and microservices.

IV. RESULT AND DISCUSSION

This section presents the results of the LRS. The first subsection describes the results in a general manner. Subsequently, the results for each question are specified in more detail.

After quality assessment, the search generated 965 sources, and 163 were obtained. The selected databases were ACM, IEEE XPLORE, SCIENCEDIRECT, SCOPUS, SPRINGER, and WEB of SCIENCE (WOS), Fig. 3.

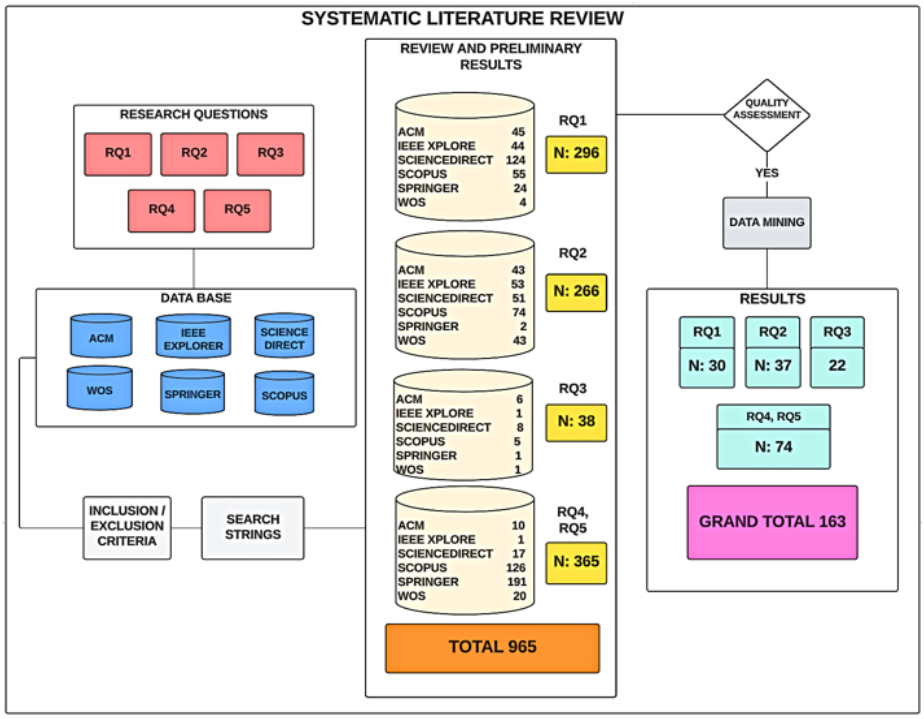

Figure 3. Results of the LRS process.

1) Q1: What are the quality attributes applicable to microservices?

For RQ1, 30 out of 296 publications were considered. The results indicate that the most frequently reported attributes for microservices are performance, scalability, security, maintainability, and coupling. In addition, with a lower frequency are cohesion, granularity, reliability, elasticity, flexibility, complexity, and interoperability. Table IV specifies the sources and frequencies of the most reported quality attributes.

TABLE IV. QUANTITATIVE SUMMARY OF THE SEARCH, EVALUATION, AND EXTRACTION OF ATTRIBUTES (RQ1)

Attribute	N⁰	References
Performance	14	[1, 3, 4, 10, 15, 19–27]
Scalability	13	[1–4, 10, 15, 19, 21, 22, 24, 27–29]
Security	10	[1, 2, 4, 21, 22, 25, 27–30]
Maintenance	9	[3, 4, 10, 20, 26, 23, 31–33]
Coupling	8	[1, 3, 14, 19, 31, 32, 34, 35]

Fig. 4 shows that 38 attributes were collected, and 17 were reported in a single study. In order to extend the analysis of RQ1, it is essential to mention the following studies:

In study of Osses and Márquez *et al.* [13], most architectural patterns and tactics were associated with five quality attributes: flexibility, testability, elasticity, performance, and scalability, with the last two being the dominant ones.

Cojocaru and Oprescu *et al.* [14] identifies a comprehensive set of quality criteria used for both Micro-Service based Architectures (MSA) and Service Oriented Architecture (SOA), the inclusion of each identified quality attribute to a recommended minimum set of quality attributes based on the significance of the decomposition context and implementation feasibility was also discussed, within these quality criteria are: granularity, coupling, cohesion, cost and response time.

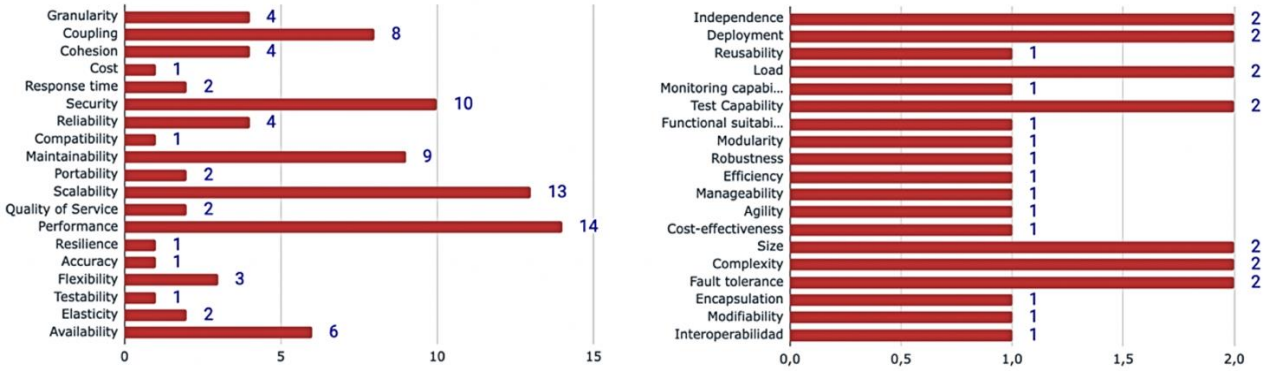

Figure 4. Results of the search process (RQ1).

Cojocaru and Uta *et al.* [35] provided a microservice validation framework that tested the quality of MS resulting from semi-automatic decomposition processes, emphasizing quality attributes such as granularity, coupling, and cohesion.

2) RQ2: What are the quality metrics applicable to microservices?

For RQ2, 37 of 266 bibliographic sources were considered. According to the LRS, the metrics applicable to the MS are cohesion, coupling, granularity, performance, and complexity, Table V.

TABLE V. QUANTITATIVE SUMMARY OF SEARCH, EVALUATION, AND EXTRACTION OF METRICS (RQ2)

Metrics	N	References
Performance	14	[3, 29–39]
Cohesion	8	[3, 23, 37, 40–46]
Coupling	8	[3, 33, 36, 39–43, 47]
Granularity	4	[14, 33, 38, 41]
Complexity	4	[3, 25, 42, 43]

The following are some interesting facts related to this question:

Zhong and Zhang *et al.* [47] determined the following results: 1) They proposed microservice coupling (MCI) measures based on the theory of relative measurement. These measures were effective in assessing the coupling between microservices. 2) They found a strong correlation between MCI coupling values and normalized change impacts on microservice pairs. It suggests that MCI coupling measures indicate the effects that change in microservices can have. Thus, it demonstrated that MCI coupling measures are constructively valid and can be used to evaluate microservice coupling in software projects. Additionally, it implies that these measures are useful and reliable for analyzing the relationship and interdependency between microservices. 3) The authors recommended further empirical evaluations in industrial microservices projects to improve the understanding of the effectiveness of MCI coupling measures in change impact analysis and microservices architecture independence. Hence, more research and experimentation are needed to verify and strengthen the usefulness of MCI coupling measures in a real microservices development context.

Other metrics appear according to the following studies:

The metrics applicable to microservices and cloud-oriented applications were elasticity, availability, isolation (performance isolation), and operational risk [45].

The microservice metrics were time (performance efficiency), percentage of accepted requests (interoperability), number of files to modify (maintainability), and energy consumption (performance efficiency) [10].

Jin and Liu *et al.* [45] used dependency metrics that were collected as time series and system metrics obtained from the underlying operating system to report the resource usage of a component and are generally related to the hardware resources on a host, including central processing unit memory, network, and disk I/O usage. It also raises the existence of application-level metrics aggregated by developers, such as the number of users, response time to a request, and execution time.

Tamburri and Bersani *et al.* [48] designated an MVC architecture pattern that used Chidamber and Kemerer (CK) metrics [49]. The Lack of Cohesion in Methods (LCOM) measures how tightly linked the internal elements of a software module are. The Number of Root Classes (NOR) measures how many class hierarchies there are in the program, Depth of Inheritance Tree (DIT) measures the length of a class hierarchy, response for a class (RFC) measures the number of methods and constructors invoked by objects of a class, and coupling between objects (CBO) measures how many methods and instance variables of class B use the methods of class A (bidirectional uses are considered only once, inheritance-related connections are not considered). WMC (weighted methods per class) measures the sum of the cyclomatic complexity of methods in a class. Class Dependency (CD) metrics, cyclomatic measures (number of cyclomatic class dependencies), Dcy measures (number of dependencies), and Dcy* measures (number of transitive dependencies).

The LRS finds 17 metrics for microservices, the most important of which is performance. Cohesion and coupling are static analysis metrics; most metrics do not have high frequency; they are reported in three, two, and one studies.

Fig. 5 projects a broader picture of the metrics encountered.

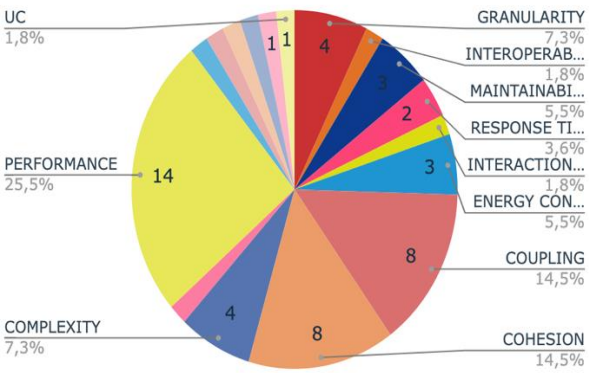

Figure 5. Quality metrics applicable to Microservices (RQ2).

3) RQ3: What challenges are identified in evaluating microservices quality?

For RQ3, 22 sources were selected out of the 38 identified, most of which relate to security, granularity, performance, monitoring, organizational strategy, orchestration, choreography, scalability, decomposition, and monolith refactoring. Some relevant work related to this issue is presented below. Additionally, Table VI lists the challenges per study. Waseem and Liang *et al.* [3] raises two research opportunities. The first is to conduct empirical studies on exploring specific tactics for microservices-based application development or analyzing trade-offs between different quality assurances (attributes). The second is constantly expanding the review to include new perspectives, such as emerging quality assurances, tactics, countermeasures, and relevant tools.

TABLE VI. CHALLENGES IDENTIFIED BY AUTHORS (RQ3)

References	Challenges
[52]	Performance-based testing, monitoring, and modeling of microservices. Particular challenges include efficient performance regression testing strategies, performance monitoring under continuous software changes, and performance modeling concepts appropriate for changing use cases.
[27]	Measurement, controlling, and maintaining a satisfactc architecture quality.
[59]	Feature determination, variability modeling, variable microservice architectures, interchangeability, customization, and re-engineering.
[26]	Lack of tools or frameworks to select third-party artifacts such as security or the latest version and lack of practitioners' knowledge of systematic methods are the main challenges that can be addressed in future work. According to the survey results, optimizing security, response time, and performance are higher priorities than resilience, reliability, fault tolerance, and memory usage.
[60]	Monitoring and adaptation mechanisms to address the diversity of microservice quality attributes, identify and resolve potential conflicts between the quality requirements of individual microservices and those of the overall application, and reconcile the adaptation needs of individual microservices and the overall application.
[61]	Infrastructure Orchestration.
[62]	Manage business processes that expand beyond the boundaries of an individual microservice.
[63]	Performance.
[64]	Architecture exploitation.
[53]	Lack of relevant skills, reliance on software as a service, organizational culture and structure, governance, difficulties associated with monolith decomposition/refactoring, master data management, orchestration, choreography, testing, and performance.
[65]	Determining microservices granularity.
[66]	Security mechanisms.
[67]	Scalability.
[54]	There is insufficient evidence on how to assess whether decisions made regarding microservices architectures produce technical debt when new system requirements need to be satisfied, e.g., high availability.
[2]	Define the level of granularity of microservices, modularization and refactoring of services, integration with the user interface, security, orchestration, monitoring, management and supervision of microservices, fault tolerance, recovery, and self-repair of microservices.
[68]	Content, mapping, tools, and business-related challenges.
[69]	Finding the proper service cut-off, decomposition, lack of expertise, DevOps and automation, organizational techniques (mindset change, cross-team collaboration, staffing, administrative and cost justification) was a major technical challenge, along with building the necessary expertise with the new technologies. Organizational challenges were especially related to large and traditional companies that simultaneously established agile processes. Initiating a change of mindset and ensuring smooth collaboration between teams was critical in these cases.
[70]	Microservice-client interactions, microservice interactions, database transactions by microservices, service discovery, fault tolerance, monitoring, logging, security, integration and deployment.
[71]	Ensuring data consistency in the system.
[72]	Heterogeneity, coupling, maintainability, security, routine testing, consistency, performance, business details.
[73]	Reliability.
[74]	Microservice container security, vulnerability management, digital research and container alternatives.
[75]	The main challenge remains modeling a function block as a service, adopting service-oriented concepts such as service-oriented architecture or microservices architecture requires addressing challenges such as service granularity or decomposition.

Valdivia and Lora-González *et al.* [10] points out that the lack of a clear definition of the interrelationship between quality attributes, metrics, and patterns in microservices is of concern for software engineering because this understanding is fundamental for correctly constructing systems based on this emerging architecture. As a result, greater clarity on this topic is needed to ensure that systems using microservices are designed and developed correctly.

Filho and Pimentel *et al.* [15] raises resilience, performance, and evolution as software architecture challenges.

The challenges in microservice development were measuring, controlling, and maintaining a satisfactory level of system architecture quality [27].

Vale and Correia *et al.* [50] provides a report on the design and execution of an interview study, with findings that support hypotheses that the body of knowledge on quality assurance in microservices architectures may be incomplete or inaccurate. It provides a greater understanding of pattern-affected quality controls, which may serve as a valuable guide for strategic engineering decision-making on moving forward with microservices adoption. Also, it points out that studies specifically targeting quality attributes in microservices are urgently needed. In addition, a systematic review of the empirical literature on microservices that adds insight into trade-offs would help researchers gain an updated perspective on the design trade-offs inherent in this architectural style.

Hassan and Bahsoon *et al.* [51] addressed the challenges of dependency on software as a service, organizational culture, and structure, governance, difficulties associated with monolith decomposition/refactoring, master data management, orchestration and choreography, testing, and performance.

According to Heinrich *et al.* [52], several challenges are highlighted concerning microservices. Generally, issues such as testing, monitoring, and performance modeling are found. In addition, particular challenges are presented, such as implementing strategies for performance regression testing, monitoring constantly evolving software environments, and using performance modeling concepts suitable for changing use cases.

Baškarada and Nguyen *et al.* [53] highlighted the challenges associated with the decomposition and refactoring of monoliths.

Marquez and Astudillo [54] indicated that there is insufficient evidence on how to evaluate whether architectural decisions made in microservices produce technical debt.

Sotomayor and Allala *et al.* [55] compare various tools that can be used for testing at different levels and present an open-source Ridesharing microservices reference prototype with which a testbed was used to evaluate new testing techniques and tools. Some results of this work express the following: 1) microservices researchers are looking for solutions to improve the correctness of microservices applications using formal techniques, 2) researchers are developing new testing techniques and tools that focus on the added complexity of data communication required between various services, 3) it is highlighted that researchers are interested in comparing different microservices testing tools and measuring the overhead for testing microservices applications. Researchers generally look for solutions to address the unique challenges associated with microservices architecture and test these applications.

For Bogner and Wagner *et al.* [56], there are a variety of metrics for evaluating conventional systems that may also apply to microservices; however, evaluation approaches specific to microservices are rare, and automatic evaluation methods are lacking.

Rademacher and Sorgalla *et al.* [57] addresses the challenges of designing Domain-based microservices and how model-driven design can help overcome them. These challenges include defining appropriate boundaries for microservices, managing communication and consistency between them, maintaining domain model consistency, and enabling independent evolution and scalability. The Model-driven design addresses these challenges by providing a clear view of the domain and the relationships between components, helping to define communication contracts, maintaining model consistency, and identifying entry points and interfaces for microservice evolution and scalability.

It is also important to highlight the study [58], which aimed to investigate the differences between ideal theoretical visions and actual industry practices related to microservices and analyze the costs that microservices imply for industry professionals. They interviewed practitioners from 20 software companies; the results identified eight pairs of common practices and challenges. They also identified five research areas that require further exploration based on challenges from the practitioners' perspective: systematic evaluation and assessment, organizational transformation, decomposition, distributed monitoring, and troubleshooting.

In summary, the most critical challenges related to microservices quality research are as follows:

The lack of specific assessment approaches to effectively measure microservices' quality requires additional research to develop automatic and specific assessment methods.

Knowledge about quality assurance in microservices architectures is incomplete, so additional research is needed to understand better the quality controls affected by patterns in microservices.

There is a lack of sufficient evidence to assess the impact of architectural decisions on microservice quality and how to address the resulting technical debt, requiring studies that develop methods and approaches to addressing this problem.

Issues of clarity in the interrelationship between quality attributes, metrics, and patterns in microservices also require additional research to improve understanding of this relationship. Need for empirical studies on specific tactics and quality trade-offs to understand how these decisions affect microservices performance and development.

Overall, these challenges and research gaps underscore the importance of additional research and exploration in the field of microservices quality in order to improve understanding and practices in this evolving area.

4) RQ4: What are the most commonly used microservices evaluation tools?

Microservices evaluation tools were reported in 74 out of 365 sources. Most studies focus on implementing or investigating a particular tool with which they perform the evaluations and apply the quality metrics. In this context, more than 70 different evaluation tools can be cited. Table VII below shows a sample of the tools found, the only ones cited in more than one study being "Jaeger" and "Zipking," the others being mentioned in only one study.

TABLE VII. SUMMARY OF THE USE OF ASSESSMENT TOOLS

Attribute	N	References
Jaeger	3	[76–78]
Zipking	2	[76, 77]
Grafana	1	[79]
Kaiju	1	[77]
Prometheus	1	[77]
Opentracing	1	[77]
Aroma	1	[80]
MAAT	1	[81]
MSA Nose+	1	[82]
Prevant	1	[83]
Docker Compose Rule	1	[55]

5) RQ5: What metrics do the tools use?

From the search conducted for RQ4, the results for RQ5 were also obtained (metrics used by the evaluation tools). Seventy-three sources were identified, and it determined that the tools use application metrics such as the number of users, response times, start time, and end time (Fig. 6) and system metrics such as CPU, memory, and disk resource usage (Fig. 7).

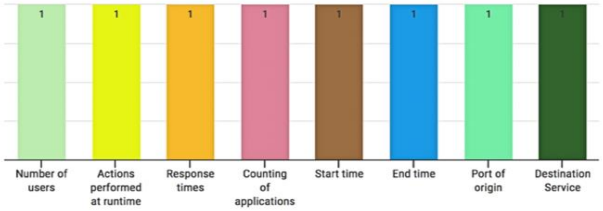

Figure 6. Application level metrics.

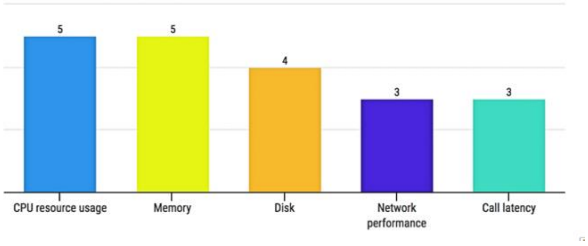

Figure 7. System or infrastructure metrics.

The tools most frequently apply the following metrics: trace, latency, and resources [36]; checking the resource usage of pods (individual instances of a running process), the number of replicas of a given service, whether any pod is in a restart cycle, or whether the developer can reschedule all pods in case of a node failure [46]; CPU performance and hard disk usage, network latency and other infrastructure metrics around the system and components [42]; start time, end time, source IP, destination service, destination instance, destination IP, destination role, whether metrics associated with the source and destination of the request are saved [59].

In the case of RQ4 and RQ5, it can be stated that the reported tools focus the evaluation on measuring runtime applications. Therefore, they use metrics that allow determining parameters such as the time in which a microservice is available and accessible for use; responsiveness and execution speed, including metrics such as response time, latency, and performance under load; the ability to handle an increase in the workload and the number of requests without degrading its performance.

V. Conclusion

In this study, the criteria, challenges, and tools related to the quality of microservices have been analyzed based on the scientific publications of recent years. The results found allow us to deduce the following:

The quality attributes applicable to microservices include performance, scalability, security, maintainability, and coupling. Among them, performance is the attribute of most concern and research. In terms of metrics (RQ2), it is observed that performance metrics are the most cited, followed by cohesion, coupling, granularity, and complexity.

Several tools were identified for evaluating microservices. Among the metrics used by these tools, several common ones were found, such as traceability, latency, CP resource usage, and infrastructure metrics related to the system and components. In addition, some application metrics, such as number of users, actions performed during execution, and response times, are mentioned. It is important to note that most tools evaluate microservices' performance metrics.

There are several challenges identified when evaluating the quality of microservices. These challenges include security, granularity, performance, monitoring, organizational strategy, orchestration, choreography, scalability, decomposition, and monolith refactoring. Several research opportunities are also mentioned, such as conducting empirical studies on specific tactics for microservices-based application development, analyzing trade-offs between different quality guarantees, and constantly expanding the review to include new perspectives and relevant tools.

In addition, challenges related to the lack of a clear definition of the interrelationship between quality attributes, metrics, and patterns in microservices, automatic evaluation of microservices, evaluation of architectural decisions in terms of technical debt, domain-based microservice design, testing, monitoring and performance modeling, among others, are highlighted. Challenges related to orchestration, communication management, consistency between microservices, independent evolution, and scalability are also mentioned.

Overall, the need to address the unique challenges associated with microservices architecture and testing of these applications is highlighted, and the importance of further research to improve understanding and practices related to microservices quality.

These challenges and research gaps highlight the need for continued exploration and study in microservices quality to improve understanding and practices in this emerging area of software architecture.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Verónica C. Tapia conducted the literature review, analyzed the data, and wrote the article; Carlos M. Gaona conducted the research and reviewed the different versions of the paper; both of the two authors approved the final version.

References

[1] S. Li *et al.*, "Understanding and addressing quality attributes of microservices architecture: A Systematic literature review," *Inf. Softw. Technol.*, vol. 131, 106449, 2021. doi: 10.1016/j.infsof.2020.106449

[2] F. H. Vera-Rivera, C. M. Gaona Cuevas, and H. Astudillo, "Development of microservice-based applications: Trends and research challenges," *Rev. Ibérica Sist. e Tecnol. Informação*, vol. E23, pp. 107–120, 2019. (in Spanish)

[3] M. Waseem, P. Liang, and M. Shahin, "A systematic mapping study on microservices architecture in DevOps," *J. Syst. Softw.*, vol. 170, 110798, 2020. doi: 10.1016/j.jss.2020.110798

[4] J. A. Valdivia, X. Limon, and K. Cortes-Verdin, "Quality attributes in patterns related to microservice architecture: A systematic literature review," in *Proc. 2019 7th Int. Conf. Softw. Eng. Res. Innov. CONISOFT 2019*, 2019, pp. 181–190. doi: 10.1109/CONISOFT.2019.00034

[5] A. Razzaq and S. A. K. Ghayyur, "A systematic mapping study: The new age of software architecture from monolithic to microservice architecture—Awareness and challenges," *Comput. Appl. Eng. Educ.*, vol. 31, no. 2, pp. 421–451, 2023. doi: 10.1002/cae.22586

[6] M. Milić and D. Makajić-Nikolić, "Development of a quality-Based model for software architecture optimization: A case study of monolith and microservice architectures," *Symmetry (Basel).*, vol. 14, no. 9, 2022. doi: 10.3390/sym14091824

[7] ISO 25000. ISO 25OOO Calidad de software y datos. [Online]. Available: https://iso25000.com/index.php/normas-iso-25000

[8] S. Jeon, M. Han, E. Lee, and K. Lee, "Quality attribute driven agile development," in *Proc. 2011 9th Int. Conf. Softw. Eng. Res. Manag. Appl. SERA 2011*, 2011, pp. 203–210. doi: 10.1109/SERA.2011.24

[9] G. Arcos-Medina and D. Mauricio, *Aspects of Software Quality Applied to the Process of Agile Software Development: A Systematic Literature Review*, vol. 10, no. 5. Springer India, 2019.

[10] J. A. Valdivia, A. Lora-González, X. Limón, K. Cortes-Verdin, and J. O. Ocharán-Hernández, "Patterns related to microservice architecture: A multivocal literature review," *Program. Comput. Softw.*, vol. 46, no. 8, pp. 594–608, Dec. 2020. doi: 10.1134/S0361768820080253

[11] T. Schirgi and E. Brenner, "Quality assurance for microservice architectures," in *Proc. IEEE Int. Conf. Softw. Eng. Serv. Sci. ICSESS*, 2021, pp. 76–80. doi: 10.1109/ICSESS52187.2021.9522227

[12] C. Richardson. Microservice architecture. [Online]. Available: https://microservices.io/patterns/microservices.html

[13] F. Osses, G. Márquez, and H. Astudillo, "An exploratory study of academic architectural tactics and patterns in microservices: A systematic literature review," in *Proc. Av. en Ing. Softw. a Niv. Iberoam. CIbSE 2018*, 2018, pp. 71–84.

[14] M. D. M.-D. Cojocaru, A. Oprescu, and A. Uta, "Attributes assessing the quality of microservices automatically decomposed from monolithic applications," in *Proc. 2019 18th Int. Symp. Parallel Distrib. Comput. ISPDC 2019*, no. 1, 2019, pp. 84–93. doi: 10.1109/ISPDC.2019.00021

[15] M. Filho, E. Pimentel, W. Pereira, and P. Maia, "Self-adaptive microservice-based systems—Landscape and research opportunities," in *Proc. 2021 International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 2021. doi: 10.1109/SEAMS51251.2021.00030

[16] T. Gorski and A. P. Wozniak, "Optimization of Business process execution in services architecture: A systematic literature review," *IEEE Access*, vol. 9, pp. 111833–111852, 2021. doi: 10.1109/ACCESS.2021.3102668

[17] R. Capuano and H. Muccini, "A systematic literature review on migration to microservices: A quality attributes perspective," in *Proc. 2022 IEEE 19th International Conference on Software Architecture Companion (ICSA-C)*, 2022. doi: 10.1109/ICSA-C54293.2022.00030

[18] B. Kitchenham, "Procedures for performing systematic reviews," Keele Univ. Tech. Rep. TR/SE-0401, 2004.

[19] F. H. Vera-Rivera, H. Astudillo, and C. M. Gaona-Cuevas, "Defining and measuring microservice granularity—A literature overview," *PeerJ Computer Science*, vol. 20, 2017.

[20] A. Yamuç and U. M. Sürme, "A service-oriented architecture to microservice architecture migration strategy for satellite ground software systems ," in *Proc. 7th Turkish National Software Architecture Conference, UYMK 2018*, 2018, vol. 2291, pp. 8–19.

[21] S. A. K. A. K. Ghayyur, A. Razzaq, S. Ullah, and S. Ahmed, "Matrix clustering based migration of system application to microservices architecture," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 1, pp. 284–296, 2018. doi: 10.14569/IJACSA.2018.090139

[22] I. Ranawaka *et al.*, "Custos: Security middleware for science gateways," in *Proc. Practice and Experience in Advanced Research Computing*, 2020, pp. 278–284. doi: 10.1145/3311790.3396635

[23] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: A systematic mapping study," *J. Syst. Softw.*, vol. 150, pp. 77–97, 2019. doi: 10.1016/j.jss.2019.01.001

[24] S. Eismann, C.-P. Bezemer, W. Shang, D. Okanović, and A. van Hoorn, "Microservices: A performance tester's dream or nightmare?" in *Proc. the ACM/SPEC International Conference on Performance Engineering*, 2020, pp. 138–149. doi: 10.1145/3358960.3379124

[25] F. Klinaku, D. Bilgery, and S. Becker, "The applicability of palladio for assessing the quality of cloud-based microservice architectures," in *Proc. the 13th European Conference on Software Architecture*, vol. 2, 2019, pp. 34–37. doi: 10.1145/3344948.3344961

[26] J. Ghofrani and D. Lübke, "Challenges of microservices architecture: A survey on the state of the practice," in *Proc. CEUR Workshop*, vol. 2072, 2018, pp. 1–8.

[27] M. Cardarelli, A. Di Salle, L. Iovino, I. Malavolta, P. Di Francesco, and P. Lago, "An extensible data-driven approach for evaluating the quality of microservice architectures," in *Proc. 34th Annual ACM Symposium on Applied Computing, SAC 2019*, 2019, vol. 1477, pp. 1225–1234. doi: 10.1145/3297280.3297400

[28] J. Bogner, J. Fritzsch, S. Wagner, and A. Zimmermann, "Microservices in industry: Insights into technologies, characteristics, and software quality," in *Proc. 2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*, 2019, pp. 187–195. doi: 10.1109/ICSA-C.2019.00041

[29] M. Stocker, O. Zimmermann, U. Zdun, D. Lübke, and C. Pautasso, "Interface quality patterns—Communicating and improving the quality of microservices APIs," in *Proc. the 23rd European Conference on Pattern Languages of Programs*, 2018. doi: 10.1145/3282308.3282319

[30] G. Márquez and H. Astudillo, "Identifying availability tactics to support security architectural design of microservice-based systems," in *Proc. 13th European Conference on Software Architecture, ECSA 2019*, vol. 2, 2019, pp. 123–131. doi: 10.1145/3344948.3344996

[31] M. Gao, M. Chen, A. Liu, W. H. Ip, and K. L. Yung, "Optimization of microservice composition based on artificial immune algorithm considering fuzziness and user preference," *IEEE Access*, vol. 8, pp. 26385–26404, 2020. doi: 10.1109/ACCESS.2020.2971379

[32] J. Bogner, J. Fritzsch, S. Wagner, and A. Zimmermann, "Limiting technical debt with maintainability assurance: An industry survey on used techniques and differences with service- and microservice-based systems," in *Proc. the 2018 International Conference on Technical Debt*, 2018, pp. 125–133. doi: 10.1145/3194164.3194166

[33] J. Bogner, A. Zimmermann, and S. Wagner, "Towards an evolvability assurance method for service-based systems," in *Proc. 7th European Conference on Service-Oriented and Cloud Computing, ESOCC 2018,*. 2020, vol. 1115, pp. 131–139. doi: 10.1007/978-3-030-63161-1_10

[34] J. Bogner, S. Wagner, and A. Zimmermann, "Automatically measuring the maintainability of service- and microservice-based systems - a literature review," in *Proc. 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*, 2017, pp. 107–115. doi: 10.1145/3143434.3143443

[35] M. Cojocaru, A. Uta, and A.-M. A. M. Oprescu, "MicroValid: A validation framework for automatically decomposed microservices," in *Proc. Int. Conf. Cloud Comput. Technol. Sci. CloudCom.*, 2019, pp. 78–86. doi: 10.1109/CloudCom.2019.00023

[36] J. Bogner, S. Wagner, and A. Zimmermann, "On the impact of service-oriented patterns on software evolvability: A controlled experiment and metric-based analysis," *PeerJ Comput. Sci.*, vol. 2019, no. 8, 2019. doi: 10.7717/peerj-cs.213

[37] J. Bogner, S. Wagner, and A. Zimmermann, "Using architectural modifiability tactics to examine evolution qualities of service- and microservice-based systems: An approach based on principles and patterns," *Software-Intensive Cyber-Physical Syst.*, vol. 34, no. 2–3, pp. 141–149, 2019. doi: 10.1007/s00450-019-00402-z

[38] C. Trubiani, A. Bran, A. van Hoorn, A. Avritzer, and H. Knoche, "Exploiting load testing and profiling for performance antipattern detection," *Inf. Softw. Technol.*, vol. 95, pp. 329–345, 2018. doi: 10.1016/j.infsof.2017.11.016

[39] C. Schröer, F. Kruse, and J. Marx Gómez, "A qualitative literature review on microservices identification approaches," *Commun. Comput. Inf. Sci.*, vol. 1310, pp. 151–168, 2020. doi: 10.1007/978-3-030-64846-6_9

[40] A. Selmadji, A.-D. Seriai, H. L. Bouziane, C. Dony, and R. O. Mahamane, "Re-architecting OO software into microservices: A quality-centred approach," in *Proc. 7th European Conference on Service-Oriented and Cloud Computing, ESOCC 2018*, 2018, vol. 11116, pp. 65–73. doi: 10.1007/978-3-319-99819-0_5

[41] S. Li *et al.*, "A dataflow-driven approach to identifying microservices from monolithic applications," *J. Syst. Softw.*, vol. 157, 110380, 2019. doi: https://doi.org/10.1016/j.jss.2019.07.008

[42] O. Al-Debagy and P. Martinek, "A metrics framework for evaluating microservices architecture designs," *J. Web Eng.*, vol. 19, no. 3–4, pp. 341–369, 2020. doi: 10.13052/jwe1540-9589.19341

[43] Y. Zhang, B. Liu, L. Dai, K. Chen, and X. Cao, "Automated microservice identification in legacy systems with functional and non-functional metrics," in *Proc. IEEE 17th International Conference on Software Architecture, ICSA 2020*, Mar. 2020, pp. 135–145. doi: 10.1109/ICSA47634.2020.00021

[44] A. Selmadji, A.-D. Seriai, H. L. Bouziane, R. O. Mahamane, P. Zaragoza, and C. Dony, "From monolithic architecture style to microservice one based on a semi-automatic approach," in *Proc. 17th IEEE International Conference on Software Architecture*, 2020, pp. 157–168. doi: 10.1109/ICSA47634.2020.00023

[45] W. Jin, T. Liu, Q. Zheng, D. Cui, and Y. Cai, "Functionality-oriented microservice extraction based on execution trace clustering," in *Proc. 25th IEEE International Conference on Web Services*, 2018, pp. 211–218. doi: 10.1109/ICWS.2018.00034

[46] M. Abdellatif *et al.*, *State of the Practice in Service Identification for SOA Migration in Industry*, 2018, pp. 634–650.

[47] C. Zhong, H. Zhang, C. Li, H. Huang, and D. Feitosa, "On measuring coupling between microservices," *J. Syst. Softw.*, vol. 200, 2023. doi: 10.1016/j.jss.2023.111670

[48] D. A. Tamburri, M. M. Bersani, R. Mirandola, and G. Pea, "DevOps service observability by-design: Experimenting with model-view-controller BT- service-oriented and cloud computing," in *Proc. 7th IFIP WG 2.14 European Conference*, 2018, pp. 49–64.

[49] S. R. Chidamber and C. F. Kemerer, "A metrics suite for object oriented design," *IEEE Trans. Softw. Eng.*, vol. 20, no. 11, 1994. doi: 10.1145/118014.117970

[50] G. Vale, F. F. Correia, E. M. Guerra, T. de O. Rosa, J. Fritzsch, and J. Bogner, "Designing microservice systems using patterns: an empirical study on quality trade-offs," in *Proc. 2022 IEEE 19th International Conference on Software Architecture (ICSA)*, Mar. 2022, pp. 69–79. doi: 10.1109/ICSA53651.2022.00015

[51] S. Hassan, R. Bahsoon, and R. Kazman, "Microservice transition and its granularity problem: A systematic mapping study," *Softw. Pract. Exp.*, vol. 50, no. 9, pp. 1651–1681, 2020. doi: 10.1002/spe.2869

[52] R. Heinrich *et al.*, "Performance engineering for microservices: Research challenges & directions," in *Proc. ICPE 2017, Companion 2017 ACM/SPEC Int. Conf. Perform. Eng.*, pp. 223–226, 2017. doi: 10.1145/3053600.3053653

[53] S. Baškarada, V. Nguyen, and A. Koronios, "Architecting microservices: Practical opportunities and challenges," *J. Comput. Inf. Syst.*, vol. 60, issue 5, pp. 428–436, 2020. doi: 10.1080/08874417.2018.1520056

[54] G. Márquez and H. Astudillo, "Helping novice architects to manage architectural technical debt in microservices architecture," in *Proc. 13th Ibero-American Conference on Software Engineering and Knowledge Engineering, JIISIC 2018*, 2018, pp. 97–108.

[55] J. P. Sotomayor, S. C. Allala, P. Alt, J. Phillips, T. M. King, and P. J. Clarke, "Comparison of runtime testing tools for microservices," in *Proc. Int. Comput. Softw. Appl. Conf.*, 2019, vol. 2, pp. 356–361. doi: 10.1109/COMPSAC.2019.10232

[56] J. Bogner, S. Wagner, and A. Zimmermann, "Towards a practical maintainability quality model for serviceand microservice-based systems," in *Proc. 11th European Conference on Software Architecture, ECSA 2017*, 2017, pp. 195–198. doi: 10.1145/3129790.3129816

[57] F. Rademacher, J. Sorgalla, and S. Sachweh, "Challenges of domain-driven microservice design: A model-driven perspective," *IEEE Softw.*, vol. 35, no. 3, pp. 36–43, 2018. doi: 10.1109/MS.2018.2141028

[58] X. Zhou *et al.*, "Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry," *J. Syst. Softw.*, vol. 195, 111521, 2023. doi: 10.1016/j.jss.2022.111521

[59] W. K. G. Assunção, J. Krüger, and W. D. F. Mendonça, "Variability management meets microservices: six challenges of re-engineering microservice-based webshops," in *Proc. ACM Int. Conf. Proceeding Ser.*, 2020, pp. 14–24. doi: 10.1145/3382025.3414942

[60] N. C. Mendonca, P. Jamshidi, D. Garlan, and C. Pahl, "Developing self-adaptive microservice systems: Challenges and directions," *IEEE Softw.*, 2019. doi: 10.1109/MS.2019.2955937

[61] C. T. Joseph and K. Chandrasekaran, "IntMA: Dynamic Interaction-aware resource allocation for containerized microservices in cloud environments," *J. Syst. Archit.*, vol. 111, no. May, 2020. doi: 10.1016/j.sysarc.2020.101785

[62] D. Monteiro, P. H. M. Maia, L. S. Rocha, and N. C. Mendonça, "Building orchestrated microservice systems using declarative business processes," *Serv. Oriented Comput. Appl.*, vol. 14, no. 4, pp. 243–268, Dec. 2020. doi: 10.1007/s11761-020-00300-2

[63] F. Faticanti, F. De Pellegrini, D. Siracusa, D. Santoro, and S. Cretti, "Throughput-aware partitioning and placement of applications in fog computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, 9189841, pp. 2436–2450, 2020. doi: 10.1109/TNSM.2020.3023011

[64] R. Brondolin and M. D. Santambrogio, "A black-box monitoring approach to measure microservices runtime performance," *ACM Trans. Archit. Code Optim.*, vol. 17, no. 4, 2020. doi: 10.1145/3418899

[65] F. H. Vera-Rivera, E. G. Puerto-Cuadros, H. Astudillo, and C. M. Gaona-Cuevas, "Microservices backlog—A model of granularity Specification and microservice identification BT-services computing—SCC 2020," in *Proc. 17th International Conference on Services Computing, SCC 2020, held as part of the Services Conference Federation*, 2020, vol. 12409 LNCS, pp. 85–102. doi: 10.1007/978-3-030-59592-0_6

[66] A. Pereira-Vale, G. Marquez, H. Astudillo, and E. B. Fernandez, "Security mechanisms used in microservices-based systems: A systematic mapping," in *Proc. 2019 XLV Latin American Computing Conference (CLEI)*, 2019. doi: 10.1109/CLEI47609.2019.235060

[67] G. Márquez, M. M. Villegas, and H. Astudillo, "An empirical study of scalability frameworks in open source microservices-based systems," in *Proc. 37th International Conference of the Chilean Computer Science Society, SCCC 2018*, 2018. doi: 10.1109/SCCC.2018.8705256

[68] M. Kleehaus and F. Matthes, "Challenges in documenting microservice-based IT landscape: A survey from an enterprise architecture management perspective," in *Proc. 2019 IEEE 23rd International Enterprise Distributed Object Computing Conference (EDOC)*, Oct. 2019, pp. 11–20. doi: 10.1109/EDOC.2019.00012

[69] J. Fritzsch, J. Bogner, S. Wagner, and A. Zimmermann, "Microservices migration in industry: intentions, strategies, and challenges," in *Proc. 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Sep. 2019, pp. 481–490. doi: 10.1109/ICSME.2019.00081

[70] R. M. Munaf, J. Ahmed, F. Khakwani, and T. Rana, "Microservices architecture: Challenges and proposed conceptual design," in *Proc. 2019 International Conference on Communication Technologies (ComTech)*, Mar. 2019, pp. 82–87. doi: 10.1109/COMTECH.2019.8737831

[71] J. D. Pereira *et al.*, "A platform to enable self-adaptive cloud applications using trustworthiness properties," in *Proc. 15th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS 2020*, 2020, pp. 71–77. doi: 10.1145/3387939.3391608

[72] T. Cerny *et al.*, "On code analysis opportunities and challenges for enterprise systems and microservices," *IEEE Access*, vol. 8, pp. 159449–159470, 2020. doi: 10.1109/ACCESS.2020.3019985

[73] A. Avritzer, "Challenges and approaches for the assessment of micro-service architecture deployment alternatives in DevOps: A tutorial presented at ICSA 2020," in *Proc. 2020 IEEE International Conference on Software Architecture Companion (ICSA-C)*, Mar. 2020, pp. 1–2. doi: 10.1109/ICSA-C50368.2020.00007

[74] S. Sultan, I. Ahmad, and T. Dimitriou, "Container security: Issues, challenges, and the road ahead," *IEEE Access*, vol. 7, pp. 52976–52996, 2019. doi: 10.1109/ACCESS.2019.2911732

[75] A. Homay, A. Zoitl, M. de Sousa, M. Wollschlaeger, and C. Chrysoulas, "Granularity cost analysis for function block as a service," in *Proc. 2019 IEEE 17th International Conference on Industrial Informatics (INDIN)*, Jul. 2019, pp. 1199–1204. doi: 10.1109/INDIN41052.2019.8972205

[76] M. Cristian, "Evaluation of an end-to-end testing tool for micro-services implemented in Java and Node.JS," Universidad de Costa Rica, 2020.

[77] M. Scrocca, R. Tommasini, A. Margara, E. D. Valle, and S. Sakr, "The Kaiju project: Enabling event-driven observability," in *Proc. the 14th ACM International Conference on Distributed and Event-based Systems*, 2020. doi: 10.1145/3401025.3401740

[78] Y. Gan *et al.*, "Seer: Leveraging big data to navigate the complexity of performance debugging in cloud microservices," in *Proc. International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS*, 2019, pp. 19–33.doi: 10.1145/3297858.3304004

[79] F. Pina, J. Correia, R. Filipe, F. Araujo, J. Cardroom, "Nonintrusive monitoring of microservice-based systems," in *Proc. 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, 2018. doi: 10.1109/NCA.2018.8548311

[80] P. Bacchiega, I. Pigazzini, and F. A. Fontana, "Microservices smell detection through dynamic analysis," in *Proc. 48th Euromicro Conf. Softw. Eng. Adv. Appl. SEAA 2022*, pp. 290–293, 2022. doi: 10.1109/SEAA56994.2022.00052

[81] T. Engel, M. Langermeier, B. Bauer, and A. Hofmann, "Evaluation of microservice architectures: A metric and tool-based approach,"

in *Proc. International Conference on Advanced Information Systems Engineering*, 2018, pp. 74–89.
[82] S. Pulnil and T. Senivongse, "A microservices quality model based on microservices anti-patterns," in *Proc. 2022 19th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, 2022. doi: 10.1109/JCSSE54890.2022.9836297
[83] M. Schreiber, "Prevant (Preview servant): Composing microservices into reviewable and testable applications," *OpenAccess Ser. Informatics*, vol. 78, no. 5, pp. 1–5, 2020. doi: 10.4230/OASIcs.Microservices.2017-2019.5