

Service-Oriented Architecture vs. Microservices: An Empirical Comparative Analysis of Software Quality Attributes

Sandeep Sharma¹, Vijay Pal Singh²

¹Department, of Computer Engineering & Applications, Mangalayatan University, Aligarh, India

²Department, of Computer Engineering & Applications, Mangalayatan University, Aligarh, India

¹20230102_sandeep@mangalayatan.edu.in, ²drvijaypaltlotiya@gmail.com

Received: 16th Dec, 2025; Revised: 8th Feb 2026; Accepted: 12th Feb, 2026; Available Online: 28th Feb, 2026

ABSTRACT

The architectural decisions undertaken by the architects are the main factors that determine the scalability, maintainability and performance of a software system. A comparative analysis of the Service-Oriented Architecture (SOA) and Microservices Architecture (MSA) with respect to six significant software quality characteristics, namely, scalability, testability, maintainability, release cycle effectiveness, availability, and response time, is provided in the paper. The mixed-method approach is employed in this analysis integrates evidence from a systematic literature review of 25 peer-reviewed articles (2017–2025) and selected industry grey literatures (e.g., vendor white papers and engineering blogs), along with survey data from 30 industry experts working in different technical roles. The results indicate that MSA outperforms in comparison to SOA in terms of scalability, release cycle efficiency due to its modular structure, and availability and response time due to its decentralised implementation and alignment with DevOps practices. Testability and maintainability results were also context-dependent with MSA being more modular and requiring a greater amount of architectural discipline. The findings are a viable guideline for organisations considering migration from SOA to MSA.

Keywords: *Microservices, Service-Oriented architecture, Scalability, Maintainability, Software Quality Attributes, Comparative Analysis.*

How to cite this article: Sharma S, Singh VP. Service-Oriented Architecture vs. Microservices: An Empirical Comparative Analysis of Software Quality Attributes. *Int J Drug Deliv Technol.* 2026;16(2): 608-619. DOI: 10.25258/ijddt.16.2.65

I. INTRODUCTION

The architecture of modern software systems has a decisive role to play when it comes to the performance, maintainability and adaptability of software. Amid constant technological evolution organizations are forced to choose an architectural paradigm that suits their strategic goals. The decision between Service Oriented Architecture (SOA) versus Microservices Architecture (MSA) has become more and more complex influenced by the growing presence of distributed computing, cloud native applications and agile development practices. Both SOA and MSA support basic concepts like modularity, service-based integration and loose coupling; however, microservices have evolved as a more decentralized and fine-grained architectural style building upon SOA principles [24].

The focus on small independently deployable services is in a very close alignment with the current DevOps pipeline and cloud native ecosystems, which brings the benefits of faster deployment cycles, scalability, and better fault isolation. Despite these benefits, MSA presents challenges with regard to testing, service

coordination and communication between services which require careful planning and discipline in the architecture. As organisations have moved more towards cloud-based infrastructures and toward more agile development models the ramifications of architectural decisions when trying to improve software quality would become ever more pronounced. The chosen architecture has a tremendous impact on the important software quality attributes - such as scalability, testability, maintainability, efficiency of the release cycles, availability, and response time. These attributes are vital to enterprises who are looking to stay competitive in a rapidly changing technological landscape. A comprehensive understanding of the role of SOA and MSA in affecting these dimensions is therefore vital for organisations undergoing a process of digital transformation, as this enables decision-making with evidence based on long-term strategic and technical imperatives.

Existing literature offers valuable insights into discrete software quality attributes but rarely addresses the cross generalised comparisons, i.e. all dimensions of quality of software, combining the advantages and short

comings of the software architectural styles (SOA and MSA), while at the same time being comprehensive. Current research usually focuses on single dimensions, such as aspects of scalability or testability in more isolated scenarios, which are often specific to use cases or industries. Moreover, as much as MSA is often seen as a logical extension of SOA, there are unique challenges when MSA and SOA's interoperability is considered, with respect to testing, dependency management, and distributed failure handling. These challenges are particularly relevant for organisations who are trying to integrate novel MSA based systems with legacy SOA infrastructures. In this light, a systematic analysis of performance for each of these architectures along a range of software quality aspects is imperative. For example, scalability is an underlying characteristic for both of these paradigms but MSA's modular design makes it possible to scale more finely, making it more appropriate for a dynamic, cloud-centric environment. Likewise, maintainability is a crucial point, because systems must be susceptible to modification, expansion, and must be continually evolving. MSA's breaking up of services comes with maintainability benefits of being able to update services independently from each other, but in return, the modularity also requires a higher architectural discipline effort to address dependencies and allow a system to evolve seamlessly. On the other hand, the centralized nature of SOA's architecture makes it far more predictable when it comes to integration and testing, a predictable nature to its architecture because of the use of standardized protocols, and the clearly expressed interfaces. Despite this predictability, however, it usually comes at the cost of flexibility; SOA systems can face challenges in terms of scaling and updating their systems, especially in situations where rapid change is a key concern. Consequently, SOA is particularly suited for organisations that focus on stability, regulation compliance and controlled integration, i.e. financial and healthcare sectors.

This research utilises software quality models, ISO/IEC 25010 and the McCall framework, to establish metrics for evaluating software quality and comparing SOA and MSA based on performance efficiency, reliability and maintainability. Therefore, this paper will examine both types of architecture using six key metrics. It includes scalability, testability, maintainability, release-cycle efficiency, availability and response time to determine their relative advantages, disadvantages, and trade-offs. Furthermore, it utilises the findings of previous research to align empirical research with recognised constructs of software quality. This study also includes, expanding the

understanding of SOA and MSA within the context of current enterprise and cloud-native applications.

II. LITERATURE REVIEW

In recent years, wide spectrum scholarly inquiry has focused on the progression from Service-Oriented Architecture (SOA) to Microservices Architecture (MSA) paying special focus to the way the two paradigms affect important software quality attributes. Researchers and industry practitioners alike have explored the advantages and disadvantages of MSA compared to SOA and in doing so shed light on the trade-offs inherent in the adoption of either architectural model [25]. These seminal studies, supplemented by more recent studies (2017-"25), are the basis of the current analysis. Scales: Scalability is and still is the predominant advantage of MSA. Empirical investigations have proved time and again that microservices are better at scaling because of the decomposition of the service, decentralized architecture and their integration with container orchestration platforms like Kubernetes and Docker. Wang et al. [3] demonstrated the use of Spring Cloud and Kubernetes to effectively enable fine-grained scaling of individual services in contrast to SOA, in which centralised enterprise service buses (ESBs) can act as bottlenecks during certain periods of high load. Similarly, Ahmet Vedat Tokmak et al. [4] noticed that, while MSA has excellent scalability, it requires some discipline in monitoring in order to control service visibility. Industry reports, including IBM [5] emphasise the capabilities of MSA to be more suitable for cloud environments complete of dynamically fluctuating workloads.

The idea of testability in MSA is a more complicated situation. On the one hand, the modularity of MSA makes unit testing easier because loosely coupled services can be unit tested independently in the automated pipelines. On the other hand, Cerny et al. [6] argue that with distributed dependencies and asynchronous communication, integration and end-to-end testing as well as complexity are increased in MSA. In contrast, SOA makes integration testing easy because of standardised service contracts at the cost of more granular component testing. A systematic mapping study by Meijer et al. [8] further reveals that performance - designing patterns in microservices can cause an impact on latency and throughput, making testing these systems more difficult in distributed systems. About maintainability, MSA is often considered to be more beneficial because of its independent and modular structure. Tapia et al. [9] and Calderon-Gomez et al. [10] claim the existence of perceptible service boundaries in MSA that make the long term maintenance and evolution of the system

more easy. Nevertheless, Cerny et al. [6] warn that not using dependencies can lead to the of propagation of changes that are not considered, and thus can make maintains difficult. SOA with its centralised governance and standardisation lessens the chances for uncoordinated change at the expense of flexibility and responsiveness. Case studies on SOA - to - MSA migrations [11], [12], [13], [23] report increased productivity as a result of migration but also report the difficulty of maintaining architectural rigor and documentation. More recently, Mohottige et al [14] followed a systematic an in-depth review of microservices reengineering and described the two-sided advantages of scalability and adaptability and the disadvantages of increased overhead in operations and the overall complexity of the system. The release cycle is another differentiating feature between SOA and MSA. MSA supports more frequent and agile releases with its ability to provide independent deployment of services, which is a parallel of current DevOps practices. Rossi et al. [15] found that the hierarchical scaling capability of Kubernetes helps in updating systems without the need for any complete shutdowns. Meijer et al. [8] and various industry surveys [11], [12] verify that MSA saves time - to - market and boosts DevOps efficiency to allow faster feature deployments. In opposition to SOA, because it usually requires shared infrastructure and service interdependencies, it tends to require joint releases, resulting in longer deployment cycles and additional regression testing overhead.

Availability is another area in which MSA beats out SOA. Zhang et al. [16] showed that the adaptive load balancing and fault tolerant systems of MSA are responsible for the increased resilience of web systems. Rossi et al. [15] also emphasized MSA's capability of providing redundancy and automated failover mechanisms to counter the single points of failure. Conversely, the centralised ESBs of SOA introduce potential bottlenecks, which decrease the resiliency of a system. Tokmak et al. [4] also mentioned that a lack of robust monitoring frameworks in MSA can hinder the availability, especially in high-availability environments. In terms of response time or latency MSA is usually better. Lightweight communication protocols like Rest and gRPC used by microservices leads to less latency and better throughput and performance, specifically for high concurrency conditions. Meijer et al. [8] confirmed that microservices result in better response times as well but poorly design systems with too much inter service communication can cause a performance decline [17]. SOA, which often involves heavier protocols such as

the use of the SOAP protocol, does not have the low latency the web service does; however, it is more predictable in stable environments where integration is required. This evidence indicates that there are significant architectural benefits of MSA, but that service coordination and architecture design must be carefully managed to maximize the service benefits. The literature consistently shows MSA to be better than SOA across dimensions such as scalability, efficiency in the release cycle, availability, and response time. MSA also has an excellent performance in software - quality maintenance and testability under certain circumstances - yet it is intrinsically a more complicated area as a result of the interdependencies and distributed system management. SOA, while less flexible and slower in most situations, is still a viable option for systems that require centralised governance, standardised integration and predictable communication. Nonetheless, the main drawback of the research that is available is its tendency to consider individual attributes in isolation rather than offering a comprehensive and comparative analysis across the six dimensions of quality. The current study helps overcome this gap by synthesising the content of 25 peer reviewed articles, selected industry grey literature (e.g., vendor white papers and engineering blogs), and 30 industry practitioners and providing a broader, comparative view on SOA and MSA in the context of real world applications.

III. METHODOLOGY

The methodology was developed with factors that ensure rigor, transparency and practical relevance, through the systematic combination of academic evidence with practitioner insights. The comparative and empirical research design was used in this study. The main goal of this analysis is to examine SOA and MSA comparative capabilities based on six major software quality attributes: scalability, testability, maintainability, release-cycle efficiency, availability and response-time. Given the complex nature of architectural decision - making in modern software systems, a mixed methods approach has been adopted to increase the rigor and the applicability of the findings. By combining the evidence from the scholarly literature with the understanding of industry practitioners, this research offers a comprehensive picture which combines the analysis from the academy with the applications in the world. Not only is this approach better for the credibility of the results, it also ensures that the conclusions are based on the current state of the academic research and operational practices within the industry.

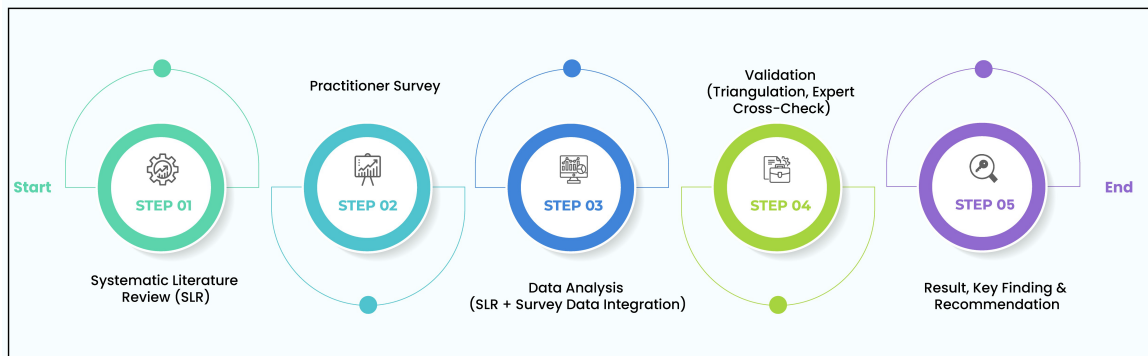


Figure 1. Methodology of Comparative Analysis of SOA and Microservices Architectures.

Figure 1 presents the overall research workflow, showing how evidence from the systematic literature review and insights from the industry survey are combined to evaluate and compare SOA and microservices across key software quality attributes.

A. Research design

The research design was chosen with the intent of providing more depth and generalisability for the results. A mixed-method design promotes triangulation, a strategy that combines various sources of information (peer-reviewed articles + selected industry grey literature + practitioner feedback) to increase the credibility of the study and reduce the possible biases in the process. Triangulation in the context of this research is done through combinations of two different and complementary sources of data:

- Systematic Literature Review (SLR)

This component is a broad summary of current academic and industry research that is related to SOA and MSA. It aims at synthesizing findings on the architectural characteristics of both of the paradigms and their impact on software quality attributes.

- Expert Survey

To validate and extend the results from the literature research, an expert survey was undertaken to obtain the knowledge of practitioners with hard-selling experience in executing and managing SOA and MSA architectures. The survey was used to inform the literature-based comparisons with real life perspectives - making the information practical and confirming theoretical outcomes.

By combining these two approaches, the research guarantees that its conclusions are based not only on

what is known from research but also on the operational realities experienced by the industry professionals. This dual approach builds a bridge between theory and practice with valuable insights for organisations approaching the problem of architectural transition.

B. Data Collection

The main objective of the systematic literature review is the identification and analysis of the body of existing literature in order to answer the research questions. The sources for this review were chosen as they relate to the relevance, credibility and the goals of this research. Specifically, the data sources are various scientific databases such as SpringerLink, IEEE Xplore, ACM Digital Library and ScienceDirect, industry reports and grey literature.

- Relevance: Sources that give direct insights into the architectural comparisons between SOA and MSA and, especially, those that deal with migration paths, architectural tradeoffs, and quality attributes of software.
- Research that is Empirical or Observational: Studies that report empirical data or observational data indicating the performance of SOA or MSA in real-life based systems, including performance measures and mitigation strategies.
- Recent Publications (2017–2025): The period 2017-2025 has been chosen because it reflects the latest change in software architecture practices, when the use of cloud-native development, container orchestration (e.g., Docker, Kubernetes) and DevOps methodologies was being widely adopted. Earlier research was not included, to prevent forward-looking research from long ago from SOA

Service-Oriented Architecture vs. Microservices: An Empirical Comparative Analysis of Software Quality Attributes

implementations for which the current architectures do not reflect real-life situations.

- **Credibility:** The selected studies were written by respected scholars or experts in the industry to ensure that the insights provided were academically rigorous practices that were applicable.

A three-stage screening process was used for ensuring the quality and relevance of the chosen literature. This process included a first title review followed by an abstract review and a final review of the full text. This method permitted the identification of 25 good sources, which consist of peer-reviewed journal articles, conference papers and a number of selected industry grey literatures. The most important findings of these researches were manually extracted and synthesised with a focus on architectural features, quality implications and real order applications related to SOA and MSA.

C. Practitioner Survey

To complement the results obtained from the literature review, a survey of practitioners was carried out in order to gain insights from professionals with direct experience in working with both SOA and MSA architectures. This survey aimed to capture the practical challenges and benefits faced by the experts working in the field so that the study will be relevant to current industry practices.

The sample of practitioners for this survey was 30 practitioners, identified from the population by purposive sampling to ensure that only individuals with a good level of and relevant experience were sampled. The respondents were from a variety of professional positions such as software architects, cloud engineers, DevOps engineers, software developers, and software support engineers. This diversity ensured that the survey data reflected a large amount of views on architectural decision-making and practical implementation problems.

- 6 Software Architects
- 6 Cloud Engineers.
- 6 DevOps Engineers
- 6 Software Developers
- 6 Software Support Engineers

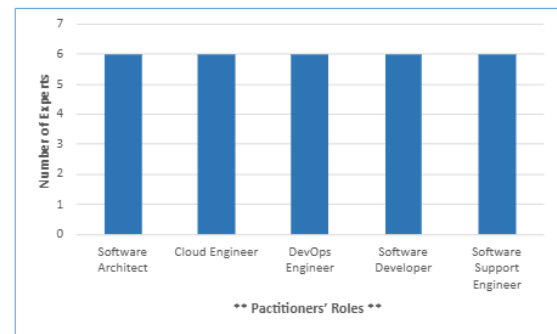


Figure 2. Distribution of Participants by Role

Figure 2 shows the distribution of participants by role. The participants were recruited using professional contacts and past work experience, and each had an active role in enterprise systems that have used both SOA and MSA architectures.

The participating experts consisted of 8 to 20 years of professional experience divided among a balance of mid-level and senior professionals. They came from a wide range of industries in finance, healthcare , e-commerce , telecommunications and software services, so they carried a broad view on architectural practices. Most participants reported working on enterprise-scale systems involving distributed, cloud-native deployments, with several also having experience in scaling mid-sized applications to production-grade environments. All the participants had hands-on experience with both Service-Oriented Architecture (SOA) and Microservices Architecture (MSA) and were able to provide learned comparative insights. All participants had practical experience working with both Service-Oriented Architecture (SOA) and Microservices Architecture (MSA), enabling them to provide informed comparative insights. The findings were based on a study of 30 domain experts who represent key professional roles such as software architects, cloud engineers, DevOps specialists, software developers and support engineers. While the sample size may seem modest, the emphasis in research to strongly represent domain knowledge via experts rather than statistical generalization drives the research. Prior studies in the evaluation of software architectures have deployed similar sample sizes of expert surveyors in order to acquire valid and consistent knowledge. The variety of experts involved and the meeting of the minds was sufficient to provide strength of evidence for credible conclusions regarding the quality attributes of SOAs

Service-Oriented Architecture vs. Microservices: An Empirical Comparative Analysis of Software Quality Attributes

and MSAs.

The people were recruited using professional contacts and previous work experiences, and to qualify as candidates they needed to have an active role in

enterprise systems that have used both SOA and MSA architectures. Table 1 below demonstrates the perceived relevance of the six software quality attributes across expert roles.

Table 1. Application of Software Quality Attributes by Expert Role.

Attribute	Software Architects	Cloud Engineers	DevOps Engineers	Software Developers	Software Support Engineers
Scalability	✓	✓	–	✓	–
Testability	✓	–	✓	✓	–
Maintainability	✓	–	✓	✓	–
Release Cycle	✓	✓	✓	✓	–
Availability	✓	✓	–	–	✓
Response Time	✓	✓	✓	✓	✓

The survey was designed to get both quantitative ratings and qualitative insights:

- **Quantitative Section:** Stakeholders rated SOA and MSA on six predefined software quality attributes on a 5-point scale. This scale provided for granular differentiation to be made in evaluations with the values ranging from 1.0 all the way to 5.0. The ratings made it possible to identify trends and variations in the way the experts rated the performance of each architecture. The questionnaire was designed based on professional positions, so that each participant only reviewed the software quality attributes known to the tested participant's expertise. As a result, attributes beyond a participant's area of expertise were not included on their survey form, and, in this way, irrelevant responses were avoided and all ratings collected were based on informed, experience-based judgments.
- **Qualitative Section:** Participants were asked to offer open-ended answers to questions relating to their professional experience with SOA and MSA. These insights created a useful context from which to shed light on the real-world challenges tied to the implementation and maintenance of each of the architectures.

D. Data Analysis

The analysis of the literature review and practitioner survey was then conducted alongside each other, and the

results were integrated into an overall comparative discourse. The results of the literature study were classified as per the 6 software quality attributes. The comparison between SOA and MSA was organised so as to identify the weaknesses and strength of both the systems in respect of these above-mentioned dimensions. Methodological neutrality and comparative integrity across attributes were established through the equal weighting of attributes. Equal weights are recommended for use within multi-criteria decision analysis (MCDA) when neither empirical nor expert-provided weights are available. The exploratory nature of the MCDA supports the use of this approach to reduce the impact of subjective bias because the importance attributed to various attributes will frequently differ based upon the specific organizational context(s) from which they were assessed. Any observed contradictions within the literature were reconciled through the process of triangulation. This was done by giving preference to the most recent empirical studies and by cross-validating the findings with those of practitioners in order to ensure that the findings are aligned with current practices within the industry.

The expert responses were gathered using a 5-point rating scale (1.0–5.0), with fractional values allowed. Therefore, the results were analysed descriptively to capture general patterns and tendencies without implying precise quantitative differences. Conflicting opinions of the experts were analysed based on mean

values to estimate the overall tendency. Attributes where greater variability was found were considered meaningfully as context-sensitive, while qualitative differences were reconciled by using thematic grouping. The qualitative feedback was organised according to themes such as testing, maintainability and deployment and the expert perception on these areas was directly compared to the literature findings. Finally, the information from the literature and the survey was merged to determine areas where the outcomes and practitioner experiences differed. This way the study was able to provide a more comprehensive view of the comparative advantages of SOA and MSA in actual use cases.

To strengthen the validity, both quantitative and qualitative results were combined using convergent triangulation in which results of both quantitative and qualitative methods were compared and reconciled. This made sure that consistency was applied to both academic and practitioner evidence to reflect both the statistical and experiential components of architectural evaluation.

E. Validation Approach

To ensure that the study was accurate and reliable, a robust validation process was implemented at every stage of the study. During the literature review three professional software architects were involved in filtering and checking the relevancy of the selected sources. This helped to minimise bias and increase the quality of the evidence. Three reviewers checked the answers to weed out any discrepancies/missing answers. The use of open-ended questions also ensured impartial analysis as it offered the participants a chance to

elaborate on their experience and offer more nuanced insights. Ethical principles, including the principles of informed consent, privacy and voluntary participation, were strictly observed throughout the study. These measures of validation ensured that the research was rigorous, fair and applicable in comparing the strengths and weaknesses of SOA and MSA.

IV. RESULTS

The findings are based on literature review and the survey of experts. All these sources were utilised simultaneously to make a comparison between Service-Oriented Architecture (SOA) and Microservices Architecture (MSA) in terms of six major software quality attributes, namely scalability, testability, maintainability, release cycle efficiency, availability, and response time. The combination of findings provides a clear picture of the performance of each architecture in this area.

A. Practitioner Survey Overview

The professional survey collected assessments of six software quality attributes derived by professionals across five roles. All in all, the findings indicate unequivocally high in favour of MSA over SOA because of the agility, modular design, and cloud-native nature of MSA as per literature.

Table 2 shows the mean ratings of SOA and MSA among six software quality attributes by experts. On the whole, MSA was ranked higher than SOA in all attributes. Certain qualities did not necessarily apply across different positions (e.g., release cycles for support engineers or testability for support roles). Therefore, role-specific survey forms were used so that all questions were applicable to each respondent.

Table 2. Practitioner Ratings of SOA and MSA on average on 6 software quality attributes.

Quality Attribute	Software Architect Avg. Rating (SOA/MSA)	Cloud Eng. Avg. Rating (SOA/MSA)	DevOps Eng. Avg. Rating (SOA/MSA)	Software Developer Avg. Rating (SOA/MSA)	Software Support Eng. Avg. Rating (SOA/MSA)	Avg. Rating (SOA/MSA)
Scalability	2.58 / 3.96	2.50 / 3.93	-	2.53 / 3.95	-	2.53 / 3.94
Testability	3.05 / 3.96	-	2.85 / 3.8	2.98 / 3.87	-	2.96 / 3.87
Maintainability	2.97 / 3.81	-	2.87 / 3.70	2.90 / 3.78	-	2.91 / 3.76
Release Cycle	2.60 / 4.23	2.61 / 4.11	2.62 / 4.19	2.55 / 4.1	-	2.59 / 4.15
Availability	2.7 / 4.11	2.61 / 4.21	-	-	2.58 / 4.15	2.63 / 4.15
Response Time	2.81 / 3.61	2.67 / 3.53	-	2.75 / 3.5	2.74 / 3.38	2.74 / 3.50

Service-Oriented Architecture vs. Microservices: An Empirical Comparative Analysis of Software Quality Attributes

To eliminate bias associated with an unequal number of representative roles, total six respondents were sampled from each of the roles. Attribute-level averages were first calculated for each of the six respondents and roles for which an attribute was not applicable were excluded from that attribute's overall mean.

Figure 3 shows a visual comparison of SOA and MSA based on average expert ratings across six software quality attributes. The chart highlights that MSA consistently outperforms SOA in all measured dimensions.

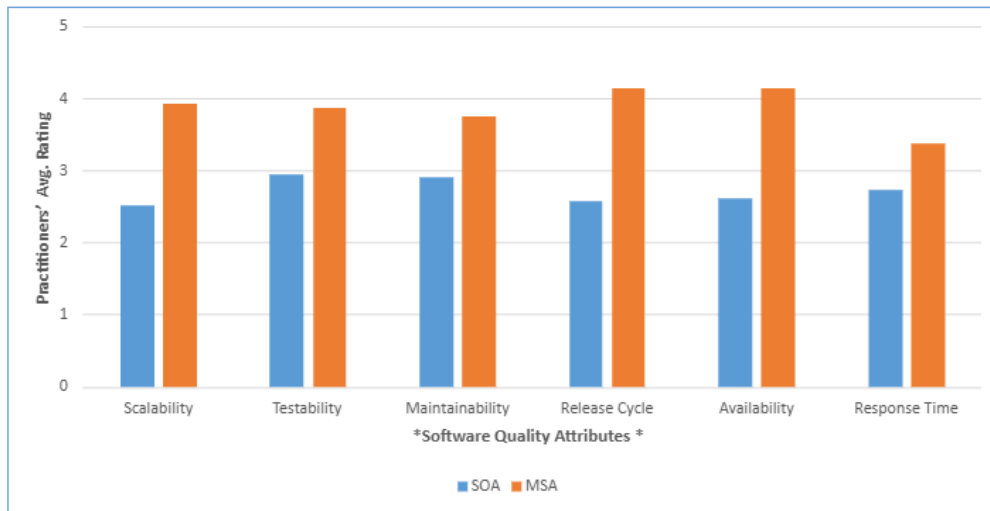


Figure 3. SOA and MSA Performance Across Six Software Quality Attributes.

Each sub-chart (Figure 4) represents one attribute, comparing role-specific perceptions of both architectures. Missing attribute indicates attributes

marked as "Not Applicable" by certain roles. The visual highlights overall trends and differences, how each role values specific quality dimensions.

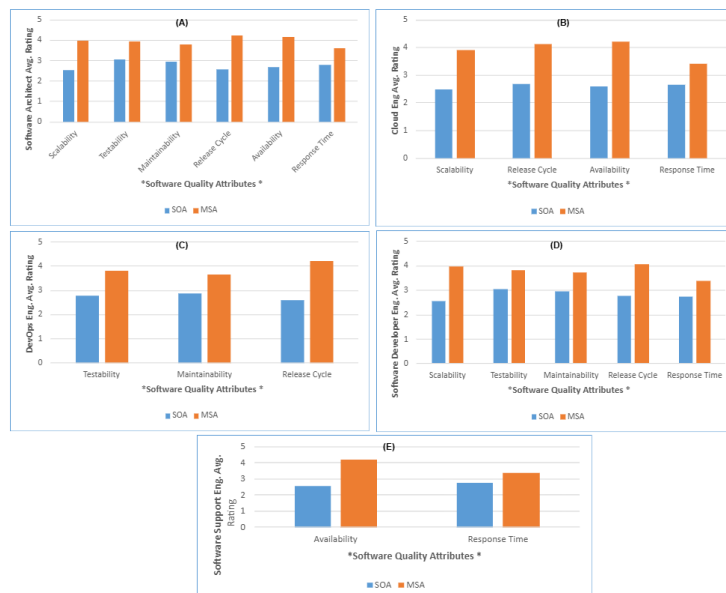


Figure 4 Expert Ratings of SOA vs. MSA across roles.

B. Attribute-Based Results

Scalability

One of the greatest differences between SOA and MSA was scalability. Literature points out that MSA enables elastic scaling by enabling individual services to scale independently using container orchestration tools including Kubernetes and Docker Swarm [3], [15][21]. It allows the dynamic distribution of resources without causing a disruption to the whole system, which makes it very efficient in the cloud-based environment with fluctuating workload.

This difference was confirmed by the expert feedback. As Table 2 indicates, MSA was rated better than SOA in the aspect of scalability. In comparison, microservices can be selectively scaled, which is more efficient and provides more agility to scalability-oriented systems.

Testability

The comparison made by testability was more subtle than that for scalability. It is indicated in literature that the modular and loosely coupled architecture of MSA allows easier unit testing [7], [13]. Integration and end-to-end testing are, however, more complicated by the nature of distributed communication, asynchronous messaging, and the fact that there is no central control [6], [7].

Table 2 illustrates that MSA was ranked superior to SOA in terms of testability. Simultaneously, participants observed that although microservices add more complexity to the testing, automation frameworks and container-based environments have mitigated many of such difficulties. They also pointed out that reliability of MSA systems on a system-wide basis would have to be orchestrated closely and monitored strictly.

Maintainability

Maintainability was another area where MSA performed well. There is a focus on literature that it is characterised by a loosely coupled and modular structure that enhances the system to modify, grow and maintainability over time [6], [13]. However, other studies caution against presence of certain latent dependencies in microservices ecosystems, which result in cascading change and complexity in general [6]. Comparatively, the governance processes and standardised interfaces of SOA are stable at a standard cost of decreased flexibility as well as delayed response to altered requirements [12].

Release Cycle

The efficiency of the release cycle is one of the aspects which MSA has clear advantages. It is mentioned in literature that it aligns with DevOps practices and continuous integration/continuous deployment (CI/CD) pipelines [8], [15], [18]. Microservices enable normal and continuous updates with reduced chances of system crash because of the self-sufficiency of services. In comparison, SOA tends to mean tightly bound deployments due to the infrastructure and dependencies, which slows down delivery and introduces complications [11].

Availability

MSA demonstrates vivid availability and fault tolerance capabilities. Literature identifies the fact that microservices that are supported by orchestration platforms have redundancy, replication, and automatic failover features [15], [16]. This type of architecture isolates faults and thus failure of one service will not send the system into a tailspin. Comparatively, centralised service buses used in SOA have single points of failure, thereby reducing resilience and availability [2], [5].

Response Time

Another aspect where MSA outperforms SOA was response time. According to the literature, microservices based on lightweight protocols such as REST and gRPC are normally associated with reduced latency and improved responsiveness even in high-concurrency settings [8], [9]. In comparison, SOA systems based on SOAP and heavy orchestration cause extra overheads thus performing sluggishly [2], [19]. Other studies warn, however, that poorly designed microservices with high inter-service communication also have poor response time as well [19].

C. Overall Interpretation

The differences between SOA and MSA ratings were analyzed using descriptive comparison of mean values of all the expert groups. Given that the study was exploratory and based on a modest sample size, formal statistical testing was not used; instead, descriptive mean comparisons were used to identify consistent directional differences between SOA and MSA ratings across roles, which is appropriate for exploratory studies focused on pattern recognition rather than statistical inference. This regular pattern indicates a possibility that the differences are meaningful and indicate actual

architectural benefits from the variations instead of random variations. Additionally, no inferential statistical tests (e.g., t-test or ANOVA) were conducted due to the exploratory nature and modest sample size of the study; therefore, the findings should be interpreted as indicative trends rather than statistically validated differences.

The literature review and the survey of experts have demonstrated mutual preference of Microservices Architecture (MSA) to Service-Oriented Architecture (SOA) in all six software quality attributes. MSA has clear benefits in terms of scalability, release cycle efficiency, availability, and response time, which can be greatly explained by its modular service design [3], [9], lightweight communication protocols [8], [9], and its good compliance with DevOps and CI/CD practices [11], [15]. Maintenance and testability also favour MSA, although other literature and expert opinion indicated difficulties including dependencies between services that are not visible [6], changing testing policies [7], and the difficulty in testing distributed systems in realistic conditions [6], [7].

V. DISCUSSION

The paper shows that Microservices Architecture (MSA) is more likely to outperform the Service-Oriented Architecture (SOA) in the six attributes that have been discussed. The literature and expert opinion were in unison regarding the advantages of MSA in scaling, release cycles, availability and responsiveness since it resonates with agile and cloud-native practices. In particular, the specialists claimed that autonomous service deployment offers greater flexibility, but research also validated the fact that it can be integrated with container orchestration and continuous delivery pipelines.

In the meantime, the outcomes demonstrate that there are serious trade-offs. The distributed nature of MSA increases complexity of integration and testing where service dependencies may cause difficulties in SOA. The higher maintainability score for MSA depends on the fact that boundaries of services must be adhered to with rigid monitoring. Failure in high maintainability score results in interdependencies compromising the long-term viability. SOA, on the other hand, with lower overall scores, is suitable in the context where regulation and interoperability are required, particularly in large enterprise systems.

This study provides a reasonable assessment by using the findings of both the scholars and practitioner values. The presence of consensus reinforces the notion of the advantages of MSA and the examples of differences can illustrate the risks that should be examined in practice, offering actionable insight for organizations planning modernization or migration projects.

A. Limitations of the Study

This work is quite useful regarding comparative analysis, but several limitations are to be noted. The specialist survey includes 30 participants from diverse roles and provides valuable practitioner insights; expanding the sample size and increasing the survey questionnaire in future studies could yield more granular, role-specific insights. The six quality attributes were considered, and the rest of the relevant aspects were not taken into consideration, such as security, interoperability and cost-efficiency. In addition, the literature review consisted of publications from 2017-2025 and, therefore, there is a chance that some earlier foundational researches have been overlooked. Moreover, as the results are based on self-reported practitioner perceptions, they should be interpreted as indicative rather than statistically generalizable.

VI. CONCLUSION AND FUTURE WORK

In this study, a comparative study of Service-Oriented Architecture (SOA) and Microservices Architecture (MSA) was conducted, based on the evidence gathered from a systematic literature review and opinions of 30 industry experts. The result is clear in that MSA generally excels in several key quality attributes such as scalability, testability, maintainability, release cycles, availability and response time than SOA. Such advantages make MSA especially good candidates for agile, cloud-native applications, where flexibility, fast deployment and scalability are very important. The modular design and decentralised nature of MSA have allowed for the agility that's mandatory in present software surroundings to listen to the evolving need of organisations. Nonetheless, the benefits of MSA are accompanied by severe problems.

The mixed-methodology entails systematic literature analysis and survey data from experts that this study has employed has greatly increased the credibility of the findings. By combining academic research with the reality of industry professionals the study pays a balanced and sound judgment of the merits and demerits

inherent in both SOA and MSA. The agreement between academic and practitioner perspectives is a strength in establishing the veracity of the conclusions, while any differences in some areas of the results provide possibilities for further inquiry. Looking To the Future research efforts could broaden from this work by involving a larger set of industry experts spanning different sectors, to provide more granular research separating the comparative advantages of SOA and MSA. In addition, adding further software quality attributes, such as security, cost-effectiveness and interoperability, might provide a more comprehensive view of the trade-offs embedded in the process of architectural choice. A more in-depth analysis of these attributes would help supplement our understanding of some of the broader implications related with the adoption of either MSA or SOA, especially in the context of modern-day cloud-based and distributed systems.

VII. REFERENCE

- [1] S. Li *et al.*, “Understanding and Addressing Quality Attributes of Microservices Architecture: A Systematic Literature Review,” *Information and Software Technology*, vol. 131, p. 106449, Oct. 2020, doi: 10.1016/j.infsof.2020.106449.
- [2] S. Ma, C.-Y. Fan, Y. Chuang, I-Hsiu. Liu, and C.-W. Lan, “Graph-based and scenario-driven microservice analysis, retrieval, and testing,” vol. 100, pp. 724–735, Nov. 2019, doi: 10.1016/j.future.2019.05.048.
- [3] Y.-T. Wang, S. Ma, Y.-J. Lai, and Y.-C. Liang, “Qualitative and quantitative comparison of Spring Cloud and Kubernetes in migrating from a monolithic to a microservice architecture,” *Service Oriented Computing and Applications*, vol. 17, no. 3, pp. 149–159, May 2023, doi: 10.1007/s11761-023-00364-w.
- [4] Ahmet Vedat Tokmak, Akhan Akbulut, and Cagatay Catal, “Boosting the visibility of services in microservice architecture,” *Cluster computing*, vol. 27, Sep. 2023, doi: 10.1007/s10586-023-04132-5.
- [5] I. C. Team, “SOA vs. microservices: What’s the difference? | IBM,” *www.ibm.com*, Apr. 19, 2024. <https://www.ibm.com/think/topics/soa-vs-microservices>
- [6] T. Cerny, M. Chy, A. Abdelfattah, J. Soldani, and J. Bogner, “On Maintainability and Microservice Dependencies: How Do Changes Propagate? [On Maintainability and Microservice Dependencies: How Do Changes Propagate?],” *Nsf.gov*, 2024, doi: 10.5220/0012725200003711.
- [7] V. Bushong *et al.*, “On Microservice Analysis and Architecture Evolution: A Systematic Mapping Study,” *Applied Sciences*, vol. 11, no. 17, p. 7856, Aug. 2021, doi: 10.3390/app11177856.
- [8] W. Meijer, C. Trubiani, and A. Aleti, “Experimental evaluation of architectural software performance design patterns in microservices,” *Journal of Systems and Software*, vol. 218, p. 112183, Dec. 2024, doi: 10.1016/j.jss.2024.112183.
- [9] F. Tapia, M. Á. Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis, “From Monolithic Systems to Microservices: A Comparative Study of Performance,” *Applied Sciences*, vol. 10, no. 17, p. 5797, Aug. 2020, doi: 10.3390/app10175797.
- [10] H. Calderón-Gómez *et al.*, “Evaluating Service-Oriented and Microservice Architecture Patterns to Deploy eHealth Applications in Cloud Computing Environment,” *Applied Sciences*, vol. 11, no. 10, p. 4350, May 2021, doi: 10.3390/app11104350.
- [11] P. Di Francesco, P. Lago, and I. Malavolta, “Migrating Towards Microservice Architectures: An Industrial Survey,” *IEEE Xplore*, Apr. 01, 2018. doi: 10.1109/ICSA.2018.00012.
- [12] Vinay Raj and R. Sadam, “Patterns for Migration of SOA Based Applications to Microservices Architecture,” *Journal of Web Engineering*, vol. 20, no. 5, Jul. 2021, doi: 10.13052/jwe1540-9589.2051.
- [13] V. Raj and K. Srinivasa Reddy, “Best Practices and Strategy for the Migration of Service-Oriented Architecture-Based Applications to Microservices Architecture,” *Algorithms for intelligent systems*, pp. 439–449, Jan. 2022, doi: 10.1007/978-981-16-7389-4_43.
- [14] Mohottige, Thakshila Imiya, A. Polyvyanyy, R. Buyya, C. Fidge, and A. Barros, “Microservices-based Software Systems Reengineering: State-of-the-Art and Future Directions,” *arXiv.org*, 2024. <https://arxiv.org/abs/2407.13915>
- [15] F. Rossi, V. Cardellini, and Francesco Lo Presti, “Hierarchical Scaling of Microservices in Kubernetes,” *Proc. IEEE Int. Autonomic Computing and Self-Organizing Systems (ACSOS)*, Aug. 2020, doi: 10.1109/acsos49614.2020.00023.
- [16] P. Zhang, L. Xiang, Z. Song, and Y. Yang, “Adaptive load balancing and fault-tolerant microservices architecture for high-availability web

systems using docker and spring cloud,” *Deleted Journal*, vol. 7, no. 7, Jul. 2025, doi: 10.1007/s42452-025-07320-7.

[17] R. Su and X. Li, “Modular Monolith: Is This the Trend in Software Architecture?,” *arXiv.org*, 2024. doi:arXiv:2401.11867.

[18] V. Velepucha and P. Flores, “A survey on microservices architecture: Principles, patterns and migration challenges,” *IEEE Access*, vol. 11, pp. 1–1, 2023, doi: 10.1109/ACCESS.2023.3305687.

[19] A. Martínez Saucedo, G. Rodríguez, F. Gomes Rocha, and R. P. dos Santos, “Migration of monolithic systems to microservices: A systematic mapping study,” *Information and Software Technology*, vol. 177, p. 107590, Oct. 2024, doi: 10.1016/j.infsof.2024.107590.

[20] K. Tuusjärvi, Jussi Kasurinen, and Sami Hyrynsalmi, “Migrating a Legacy System to a Microservice Architecture,” *e-Informatica Software Engineering Journal*, vol. 18, no. 1, pp. 240104–240104, Nov. 2023, doi: [10.37190/e-inf240104](https://doi.org/10.37190/e-inf240104).

[21] Cloud Native Computing Foundation (CNCF), “Microservices Architecture,” Cloud Native Glossary, Jun. 10, 2024. <https://glossary.cncf.io/microservices-architecture/>

[22] Amazon Web Services, “Nintendo Systems Powers Up Productivity through Platform Engineering with AWS Managed Services,” AWS Solutions Case Study, 2024. <https://aws.amazon.com/solutions/case-studies/nintendo-systems-case-study/>

[23] J. -P. Gouigoux and D. Tamzalit, “From Monolith to Microservices: Lessons Learned on an Industrial Migration to a Web Oriented Architecture,” 2017 IEEE International Conference on Software Architecture Workshops (ICSAW), Gothenburg, Sweden, 2017, pp. 62-65, doi: 10.1109/ICSAW.2017.35.

[24] Dragoni, N. et al. (2017). Microservices: Yesterday, Today, and Tomorrow. In: Mazzara, M., Meyer, B. (eds) Present and Ulterior Software Engineering. Springer, Cham. doi:[10.1007/978-3-319-67425-4_12](https://doi.org/10.1007/978-3-319-67425-4_12)

[25] G. Liu, B. Huang, Z. Liang, M. Qin, H. Zhou and Z. Li, “Microservices: architecture, container, and challenges,” 2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C), Macau, China, 2020, pp. 629-635, doi: 10.1109/QRS-C51114.2020.00107.