Risk-Oriented Analysis and Mitigation Strategies for SOA-to-MSA Migration

Sandeep Sharma[1*], and *Vijay* Pal Singh[1]

[1]Department of Computer Engineering and Applications, Mangalayatan University, Aligarh, India

Abstract. Microservices Architecture (MSA) signifies a critical approach in the efforts by the organizations to improve the agility, scalability and effectiveness of their systems in contemporary enterprise environments. Nonetheless, this change is associated with technical, organisational, and process-related challenges that introduce significant risks, which can result in project abandonment, performance degradation, or migration failure. This paper presents a comprehensive qualitative risk-oriented analysis. A systematic literature review covering the period from 2018 to 2024 and qualitative risk analysis are used to assess the impact and probability of each challenge and to create a prioritised risk matrix with an extensive mitigation structure. Key findings are highlighted as critical risks, including shared database dependencies, lack of DevOps maturity, and organisational resistance (risk score = 9) and high priority risks that include security vulnerabilities and insufficient monitoring (risk score = 6). Suggested mitigation strategies include domain-driven design, event-driven architecture, the CQRS pattern, CI/CD pipeline enhancement, and structured change management schemes. Presenting a coherent risk prioritisation model, this study provides a strong foundation for architects, technology leaders, and researchers to develop their resilient SOA-to-MSA migration strategies, thereby contributing practical value to the software engineering literature.

1 Introduction

Use The history of enterprise software architecture has been characterised by a strong transition from Service-Oriented Architecture (SOA) to Microservices Architecture (MSA), driven by the need for scalable and agile systems. Centralised orchestration and reusability of services are some characteristics of SOA that may be associated with performance bottlenecks, tight coupling, and monolithic integration levels [11]. Conversely, MSA encourages the deployment of loosely coupled and independently deployable services that align with modern DevOps workflows and continuous delivery approaches [9]. The global microservices market is projected to exceed $10 billion by 2026 [18], reflecting accelerated industry adoption. This surge underscores the urgency of understanding and managing migration risks to ensure successful transitions from SOA to MSA.

Besides its benefits, the migration from SOA to MSA is a complicated process associated with technical refactoring, alteration of structures at the organisational level, and transformation of operations. Some of the technical issues include breaking down monolithic services, coordinating distributed data, and creating a secure environment in a decentralised architecture [8]. The resistance to change, lack of skills, or other organisational barriers complicate team alignment [4], and improper DevOps pipelines and failure to monitor the deployment can hamper deployment success [3]. These obstacles pose serious threats to migration that may derail the process, consequently causing major financial and operational impacts. Although the research topic of microservices adoption has been addressed in the past [12], [13], there is a gap when it comes to risk-based and systematic analysis related to SOA-to-MSA migrations. This research fills this knowledge gap by performing a thorough analysis of migration challenges and dividing them into technical, organisational, and process dimensions and evaluating their risks in a qualitative impact-likelihood model. To achieve these objectives, a qualitative analysis model was applied, involving literature synthesis, challenge identification, risk assessment, and mitigation design. The objectives are as follows.

- Define and identify an exhaustive list of migration challenges.
- Assess the risks by their impact and probability of occurrence to prioritise mitigation activities.
- Formulate a mitigation strategy to support practitioners in achieving resilient migrations.

This article presents a domain-adapted, integrated risk prioritisation model for SOA-to-MSA migration, accompanied by comprehensive tables and diagrams, that will help architects, developers, and technology leaders plan and execute SOA-to-MSA migration. The results should help the success rate of migration and form a basis for future papers in the field of software architecture modernisation.

2 Literature Review

* Corresponding author: 20230102_sandeep@mangalayatan.edu.in

The migration between SOA and MSA has been the centre of interest in the matter of research in software engineering because of its possibilities to improve the aspects of modularity, scalability, and the speed of deployment of the system. A comprehensive literature synthesis has been developed in this section covering the period from 2018 to 2024, in three categories, i.e., technical, organisational, and process-related challenges, with emphasis on the risks and mitigation strategies. It is a review based on peer-reviewed studies and credible industry reports (e.g., Gartner, AWS, Netflix), totalling over 30 sources, providing an adequate picture of the state of knowledge.

2.1 Technical Challenges

The technical issues during SOA-to-MSA migration are associated with the architectural development between the centralised SOA systems and decentralised microservices. Dragoni et al. [8] propose an analysis of a migration of a mission-critical system with challenges in the preservation of transactional consistency and fault isolation in service decomposition. Dependent databases, the characteristic feature of SOA, indicate serious risks of data inconsistency and strict adherence [13]. To take an example, having one database shared between services may create contention issues and be hard on the scalability front, which would be difficult to refactor [12].

Another problematic issue is improper granularity of services, which in many cases leads to service sprawl or duplication of logic [5]. Too fine-grained services are bad as they are complex, and coarse-grained services obscure the advantages of MSA [9]. One more issue is security threats: distributed architectures increase the attack surface, so strong authentication and authorisation and encryption measures are required [16]. To take an example, due to the necessity to rely on API-based communication, microservices are more vulnerable to unauthorised access, which may lead to severe consequences in case they are not secured [15].

The problem with legacy system integration is also present, as in some instances, the existing components based on SOA might be based on outdated standards or middleware, which are not compatible with microservices [7]. Additionally, there are API versioning and governance concerns that introduce a mismatch between interfaces and service failures in the process [5]. These technical problems highlight the prominence of systematic refactoring strategies, including domain-driven design and event-driven architectures [13].

2.2 Organisational Challenges

The key set of factors in migration is organisational. Razzaq and Ghayyur [4] demonstrated a multi-case study that revealed the presence of coordination overhead and the absence of shared vision as the primary obstacles, especially in large enterprises where the SOA practice had already been adopted. Adoption usually has a problem in resistance to the change in the organisation, which may be instigated by cultural inertia or the fear of a cut [4]. As an example, to the extent that teams are used to a centralised governance found in SOA, they might not take well to the decentralised governance found in MSA, thereby creating a deficiency in communication and buy-in[10].

The lack of training in microservices principles is a persistent problem, and it leads to inappropriate design decisions, like faulty design of service bounds or insufficient fault tolerance [3]. Poor team ownership, in which team ownership is unclear, or even work in silos, adds fuel to the fire in terms of coordination [4]. Moreover, the use of DevOps practices vital to MSA is discouraged by culture, preventing a faster adaptation of agile practices [4]. These problems highlight the significance of migration planning in terms of change management as well as skills development.

2.3 Process-Related Challenges

The challenges based on processes focus on operational preparedness and governance. Li et al. [3] conducted a systematic literature review and highlighted pitfalls in scalability, observability and DevOps maturity. CI/CD pipelines are not fully developed or mature, which results in degraded deployment performance and cannot support the MSA continuous delivery benefits [5]. A lack of suitable monitoring and observability tools prevents the detection of faults and delays incident response in distributed systems [6]. To illustrate, troubleshooting microservices is complicated and time-consuming in the absence of a centralised logging [15].

Ponce et al. [12] stressed that the performance of the system should not be compromised because of the orchestration mechanism of SOA, but due to the service choreography one. Partial rollback plans and the absence of automated testing systems enhance the possibility of operational breakdowns and manufacturing setbacks [14]. Migrations are further complicated by inconsistent deployment environments, i.e., production and development differences [2]. These challenges associated with the processes necessitate a strong DevOps practice and governance mechanisms.

2.4 Gaps in Existing Research

Despite such extensive literature, there are a few gaps. The majority of the studies concentrate on particular matters of migration (e.g., technical patterns or case studies), ignoring the refined gathering of risk prioritisation [5]. As noted by Wang et al. [6] and Velepucha and Flores [15], observability and skill transformation are of particular importance, yet these capabilities do not detail how exactly risks should be prioritised across domains in their analyses. Moreover, the literature proposing the role of technical, organisational, and process-related issues in an entity liquidation has limited coverage on the interaction among them, which restricts its practicality to the complex base migrations [4]. This research aims to fill these gaps by incorporating a balanced set of challenges.

Whereas prior research engagements, including those presented by Razzaq and Ghayyur [4] and Fritzsch et al. [11], cover most of the migration problems, the scope of such studies usually corresponds to specific areas and there is no prioritisation system. This study answers this gap by summarising the issues into technical, organisational and process-related dimensions, implementing a scoring mechanism of impact and likelihood on the migration-specific, and developing the hierarchy of the priority risks and matching them with the mitigation strategies in the direct relationship. The overall methodology provides one usable SOA-to-MSA migration planning model and hence a unified alternative to the disparate views found in the already existing literature. Table 1 shows summary of key literature.

Table 1. Summary of key literature on SOA-to-MSA migration.

Study	Focus Area	Key Findings	Gaps Identified	Linked Risk Factors (S. No. in Table 3)
Di Francesco & Lago [10]	Industrial survey	Need for structured migration paths tailored to the organisational context.	Limited risk prioritisation	8, 12
Dragoni et al. [8]	Mission-critical system migration	Challenges in transactional consistency and fault isolation	Focus on technical issues only	1, 4, 7
Fritzsch et al. [11]	Industry practices	Knowledge gaps and fragmented migration strategies	Lack of a consolidated risk framework	2, 3, 5, 6, 15
Yarygina & Bagge [16]	Security challenges	Need for security-by-design in distributed architectures	Limited organisational focus	5, 15
Ponce et al. [12]	Rapid review of migration models	Risks of performance degradation in choreography and failures caused by inconsistent or mismatched deployment environments	Limited process-related analysis	4, 17, 18,19
Raj & Sadam [13]	Decomposition patterns	Strategies for decoupling shared databases and defining service boundaries	Limited risk assessment	1, 2
De Toledo et al. [7]	Architectural debt	Early design compromises amplify migration risks	Limited mitigation strategies	6, 7, 20
Razzaq and Ghayyur [4]	Systemic, cultural, and DevOps maturity challenges in migration	Coordination overhead, lack of shared vision, organisational resistance, and DevOps immaturity hinder migration success.	Limited technical focus	8, 9, 10, 11,12,13
Li et al. [3]	Quality attributes	Pitfalls in scalability, observability, and DevOps maturity	No risk prioritisation model	13, 14, 15, 16

The recent studies also bring quantitative and empirical sides into migration analysis. Metrics-driven case study examining the probability of changes and architectural stability of SOA-to-MSA transitions whilst offering a structured quantitative analysis of migration effect is given by Raj and Bhukya [19]. Nogueira et al. [20] also use a mixed-methods study; they survey 53 practitioners in the industry, quantifying motivations, migration activities, and challenges, specifically data consistency and testing, and thus, base the discussion of migration risk on an active practice within an industry. The combination of perspectives will supplement the qualitative synthesis in the current research and expand the range of information on priorities borrowed to focus on migration risks.

3 Methodology

This paper uses a well-designed qualitative research methodology to examine the issues and pitfalls of SOA-to-MSA migration. The methodology has been divided into multiple stages: literature review, challenge identification, risk categorisation, qualitative risk assessment and validation. The method provides a replicable and systematic analysis, which is accepted as a software engineering research standard [17].

3.1 Research Design

As a method, secondary data analysis was chosen, which provided a synthesis of results of peer-reviewed studies and reports issued by the industry between 2018 and 2024.This design exploits available empirical evidence in order to offer a global idea about migration issues without any primary data collection. Its methodology is based on a protocol of the systematic literature review, which poses transparency and rigour [17].

3.2 Data Collection

The information is obtained from academic databases (SpringerLink, IEEE Xplore, ACM Digital Library, ScienceDirect) and industry reports. It was selected based on:

- SOA-to-MSA migration relevance.
- Observational studies of problems, threats or prevention measures.
- All were published between 2018 and 2024, ensuring they are recent and relevant.

- Publication attributes of peer-reviewed or credible industry authorship.

In its second stage, three of seven screening rounds (predominantly title and abstract screening) selected approximately 30 sources, including both peer-reviewed articles and industry reports. Manual extraction was conducted based on the key information extracted, including the description of the challenges, effects of the risk, and the solutions.

3.3 Challenges Categorization by Domain

The Challenges were revealed and categorised into three domains.

3.3.1 Technical

Problems with system architecture, management and integration (e.g. use of shared databases, granularity of services).

3.3.2 Organisational

Obstacles connected to team structure, skills and culture (e.g., change refusal, skill gaps and training shortages).

3.3.3 Process-related

Areas related to practices, deployment and governance of DevOps (e.g. CI/CD maturity, monitoring).

Every challenge was directed to a particular risk, which is a situation that could affects the migration success. Each of the challenges was tailored to a specific risk that can be characterized as a situation that may influence the success of migration. Such sorting by technical, organisational and process domains provide comprehensive picture of the migration landscape. This paper does not estimate quantitatively the interrelations of the domains or examine the way risks can migrate to the other domains. It is only limited to domain-based risk identification and prioritisation.

3.4 Risk Assessment Model

Each challenge was evaluated using a qualitative scoring method along two dimensions:

- Impact: Impact of consequences in case occurrence of risk (Low, Medium, High)
- Likelihood: The estimated chances of the risk occurring (Low, Medium, High)
- Each dimension was scored on a scale from 1 to 3:
  1. Low = 1
  2. Medium = 2
  3. High = 3

The overall risk score was calculated as using Equation (1):

$$Risk\ Score = Impact \times Likelihood \quad (1)$$

The Qualitative Risk Score was determined, as mentioned, before by multiplying the Impact and the Likelihood as standard in risk assessment [1]. Although the formula itself existed before, it is used here in conjunction with a literature resulted set of SOA to MSA migration risks in areas of technical, organisational and process risks. To avoid redundancy, the scoring logic is explained only in this subsection and omitted from table narratives.

The scores are based on the levels of priority; 1 is low and 9 is critical. The risks with a score of 6 or more were picked as priority rank, per the standards of software risk management [1].

To allow clarity, 1 to 3 scale along with its compatibility with the migration risk assessment practice was applied [1]. Risks were scored by three reviewers independently (agreement 87%) and resolved by consensus. A 3×3 matrix was used to visualize the assessed risks.

Table 2 presents the qualitative risk matrix, combining impact and likelihood scores to determine overall risk severity, following the standard 3×3 scale used in qualitative risk assessment.

Table 2. Risk matrix with combined impact-likelihood scores.

-	**Low Impact (1)**	**Medium Impact (2)**	**High Impact (3)**
Low Likelihood (1)	1 (Low)	2 (Low)	3 (Medium)
Medium Likelihood (2)	2 (Low)	4 (Medium)	6 (High)
High Likelihood (3)	3 (Medium)	6 (High)	9 (Critical).

A cut-off of 6 or greater was applied, based on [1], to identify risks where impact and/or likelihood are high-those conditions that normally result in serious delays or failures in a migration. This absolute threshold contributes to the clarity of the prioritisation, but other methods (e.g. fuzzy scoring, weighted thresholds) are potentially to be analysed in follow-up research.

3.5 Risk Assessment Reliability and Sensitivity Analysis

The reliability of the coding and risk assessment process was achieved through peer review by three researchers with expertise in software architecture. Sensitivity analysis was carried out to determine the importance of using different score of impact and likelihood on the prioritisation model, which indicated the resilience of the prioritisation model. The methodology was documented in a replicable manner so that future researchers can contribute to this model.

The workflow of the SOA-to-MSA migration risk analysis applied in this research is demonstrated in Figure 1 with the steps in the sequence these stages used being literature synthesis and challenge identification, risk scoring, prioritisation, and mitigation mapping.

Fig. 1. SOA-to-MSA migration risk analysis workflow.

Although well founded in software risk management [1], the present study generalizes the impact-likelihood matrix to the context of the SOA-to-MSA migration by ensuring that technical, organisational and process risks would be included in an interrelated taxonomy and associated with a set of mitigation strategies to be discussed and vetted by experts. That domain-specific adaptation is in response to the migration-specific operational and architectural risks and not generic IT project risks.

3.6 Validation Approach

Expert review was used in validating the proposed risk prioritisation and mitigation model instead of being applied in actual case studies in industries. To ensure consistency and contextual reliability of the identified risks, scoring criteria, and associated mitigation techniques, three persons expert in the domain of software architecture and SOA-to-MSA conversion who had a significant experience in the field carried out corresponding evaluations separately. Any differences in scoring would be addressed through consensus discussions so that it would adhere to software risk management practices that are known. This process generates qualitative assurance of internal validity of the model, but not an alternative to the validation of the model under operational migration settings. This model is thus offered as theoretically sound and expert-validated model, and in the future, the real-life industrial tests will be conducted.

4 Results

A total of 20 challenges were identified in this analysis, combining previously known risks with additional ones drawn from recent literature. The problems were grouped into technical, organisational and process, and the magnitude of their effect and probability were evaluated. The findings are displayed as elaborate tables and charts to make the analysis clear and prioritised.

Table 3 maps the identified migration challenges to their corresponding domains, describes each risk, and provides impact, likelihood, and risk score values for prioritisation.

Table 3. Risk mapping of SOA-to-MSA migration challenges.

S. No	Challenge	Domain	Risk Description	Impact	Likelihood	Risk Score
1	Shared database dependencies	Technical	Data inconsistency and tight coupling	High (3)	High (3)	9
2	Improper service granularity	Technical	Redundant logic and service sprawl	High (3)	Medium (2)	6
3	Lack of API versioning/governance	Technical	Interface mismatches and integration failures	Medium (2)	High (3)	6
4	Distributed transaction handling	Technical	Inconsistent data states across services	High (3)	Medium (2)	6
5	Security vulnerabilities	Technical	Increased attack surface in distributed systems	High (3)	Medium (2)	6
6	Legacy system integration	Technical	Compatibility issues with existing SOA components	Medium (2)	Medium (2)	4
7	Inadequate fault tolerance	Technical	System failures due to poor error handling	High (3)	Low (1)	3
8	Resistance to organisational change	Organizational	Reduced buy-in and poor communication	Medium (2)	High (3)	6
9	Insufficient microservices training	Organizational	Poor design choices and implementation errors	Medium (2)	Medium (2)	4
10	Misaligned team autonomy	Organizational	Delays, ownership confusion, coordination issues	Medium (2)	Medium (2)	4
11	Lack of cross-functional collaboration	Organizational	Siloed teams leading to inconsistent service designs	Medium (2)	Medium (2)	4
12	Cultural Resistance to DevOps	Organizational	Slow adoption of agile practices	Medium (2)	High (3)	6
13	Lack of DevOps maturity	Process	Slow or unreliable CI/CD pipelines	High (3)	High (3)	9

14	Absence of service-level objectives	Process	Undefined performance and availability targets	Medium (2)	Medium (2)	4
15	Inadequate monitoring/observability	Process	Undetected failures and delayed incident response	High (3)	Medium (2)	6
16	Lack of centralised logging	Process	Troubleshooting complexity across services	Medium (2)	Medium (2)	4
17	Incomplete rollback strategies	Process	Increased risk during failed deployments	High (3)	Medium (2)	6
18	No automated testing framework	Process	Higher rate of defects in production	High (3)	Medium (2)	6
19	Inconsistent deployment environments	Process	Deployment failures due to environment mismatches	Medium (2)	High (3)	6
20	Insufficient governance frameworks	Process	Lack of policies for service lifecycle management	Medium (2)	Medium (2)	4

4.1 Analysis of Risk Severity

Represents the frequency of the risks identified based on the scoring levels of severity and domain.

Table 4. Risk severity distribution.

Risk Score	Severity	Number of Risks	Challenges (S. No)
9	Critical	2	1,13
6	High	10	2, 3, 4, 5, 8, 12, 15, 17, 18, 19
4	Medium	7	6, 9, 10, 11, 14, 16, 20
3	Low	1	**7**

Table 4 presents the identified risks spread across four levels of severity—low, medium, high, and critical.

Table 5 groups the prioritised risks by domain—technical, organisational, and process—allowing domain-level analysis of risk concentration.

Table 5. Risk domain distribution.

Domain	Number of Risks	Challenges (S. No)
Technical	7	1,2,3,4,5,6,7
Organizational	5	8,9,10,11,12
Process	8	13,14,15,16,17,18,19,20

Figure 2 presents the domain-wise distribution of identified risks, highlighting the proportion of risks in technical, organisational, and process categories based on the prioritisation results.

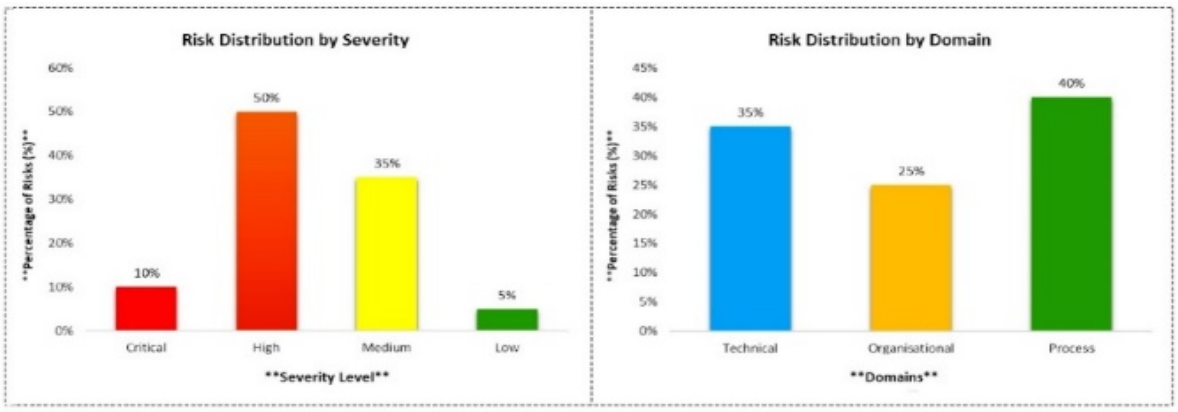

Fig. 2. Domain and severity wise distribution of risk.

4.2 Mitigation Strategies

Table 6 provides the recommended mitigation strategies of risks, which are based on the literature recommendations.

The recommended mitigation strategies of risks can be found in Table 6 on the basis of the literature recommendations. The peer reviewed and industry-based sources were used to determine the mitigation measures as they were practical. They are neither ranked nor cost-analysed and lacked empirical testing, as the present study focuses on aligning each prioritised risk with suitable mitigation options. This will be further developed in the future by considering feasibility, complexity and dependencies of the mitigations using empirical studies or simulation.

Key terms in the mitigation strategies are defined here for clarity:

- CQRS (Command Query Responsibility Segregation): A pattern of architecture that divides write and read operations in order to enhance flexibility and scalability [15].
- SAGA Pattern: A pattern of distributed transactions that ensure transactions in multiple local systems are successfully completed by a series of steps along with failure actions that are done to compensate when a step fails [13].
- Zero-Trust Architecture: A security model that confines zero-trust identity verification and least-privilege access to all users and services--wherever they are on the network [16].

Table 6. Mitigation strategies of risks.

S. No	Challenge	Mitigation Strategy
1	Shared database dependencies	Implement event-driven architectures and CQRS to decouple data stores [13].
2	Improper service granularity	Apply domain-driven design to define clear service boundaries [9].
3	Lack of API versioning/governance	Adopt API gateways and versioning standards (e.g., OpenAPI) [5].
4	Distributed transaction handling	Use saga patterns for distributed transactions [12].
5	Security vulnerabilities	Implement security-by-design and zero-trust architecture [16].
6	Legacy system integration	Use Strangler Fig pattern, API gateways, and facade layers to isolate legacy logic; integrate middleware for backward compatibility [8], [12], [14].
7	Inadequate fault tolerance	Design for resilience using circuit breakers, bulkheads, and retries; apply chaos testing to validate recovery [8], [9].
8	Resistance to organisational change	Conduct change management workshops and communicate MSA benefits [4].
9	Insufficient microservices training	Develop training programs on microservices and DevOps; conduct workshops, certifications, and code-pairing sessions [4], [5].
10	Misaligned team autonomy	Form cross-functional, service-aligned teams with clear responsibilities; apply Spotify model or team topologies for governance [4], [5].
11	Lack of cross-functional collaboration	Foster collaboration via shared KPIs, integrated sprint planning, and DevOps tooling (Jira, Confluence); run regular cross-team syncs [4], [5].
12	Cultural Resistance to DevOps	Foster a DevOps culture through training and pilot projects [4].
13	Lack of DevOps maturity	Invest in CI/CD tools (e.g., Jenkins, GitLab) and containerization [5].
14	Absence of service-level objectives	Establish SLOs and SLIs for each service; use observability tools (Prometheus, Grafana) to measure and enforce performance targets [5], [6].
15	Inadequate monitoring/observability	Deploy observability tools (e.g., Prometheus, Grafana) [3].
16	Lack of centralised logging	Use centralized logging tools (e.g., ELK Stack, Fluentd) to consolidate logs across services for efficient monitoring and debugging [3], [6].
17	Incomplete rollback strategies	Implement canary releases and blue-green deployments [14].
18	No automated testing framework	Adopt automated testing tools (e.g., JUnit, Selenium) [5].
19	Inconsistent deployment environments	Use infrastructure-as-code (e.g., Terraform) for environment consistency [5].
20	Insufficient governance frameworks	Define and enforce service governance policies, including standardized contracts, API versioning, and compliance monitoring [5].

4.3 Key Findings

4.3.1 Critical Risks (Score = 9)

Dependency on shared databases and the absence of DevOps maturity are the most serious hazards, as their systemic visions affect the consistency of data and the reliability of deployment [13], [5].

4.3.2 High-Priority Risks (Score = 6)

There are ten challenges, such as a lack of favorability of service granularity, security gaps, and poor monitoring, which must be mitigated as soon as possible, as they have the potential to derail the process of migration [12],[16],[6].

4.3.3 Medium Risks (Score = 4)

Seven risks, including the lack of training and centralised logging, are less severe but significant when it comes to success in the long term [4],[6].

4.3.4 Low Risk (Score = 3)

The risk of inadequate fault tolerance has a high impact but a low probability and should be ranked as a low priority [1],[8].

5 Discussion

It is important to note that these findings are a reinforcement of the multipronged character of SOA-to-MSA migration, that the multiple technical, organisational, and process-based challenges result in a multifaceted risk environment. A Critical technical risk (shared database dependencies, score = 9) recounts the complexity of the switch between centralised data models and distributed data stores in SOA and MSA, respectively [13]. This is in agreement with Dragoni et al. [8], who pointed out that data decoupling may result in inconsistent states when performed improperly, thus affecting performance negatively. The necessary mitigation strategies to deal with this risk include event-driven architectures and CQRS pattern [9].

The risks related to the process, in particular, the absence of DevOps maturity (score = 9), point to the change in operations needed with MSA. In sharp contrast to the centralised orchestration of SOA, MSA requires well-developed CI/CD pipelines, observability, and automation of testing [5]. Process risks (40%) are especially high, indicating that companies tend to underestimate the complexity of microservices, causing the failure of deployment or the presence of hidden problems [3]. Such tools as Prometheus and Grafana to address the issue of observability, and Jenkins to handle the risks of CI/CD, play a crucial role in addressing these risks [14].

The human element of migration finds its manifestation in such organisational challenges as resistance to change and cultural resistance to DevOps (both score = 6). Razzaq and Ghayyur [4] observed that these problems have been worsened by the existence of entrenched SOA practices and the absence of a shared vision, especially in large businesses. Training and pilot projects are required as change management strategies to encourage buy-in and align teams with MSA in having a decentralised model [4].

The risk prioritisation model gives the practitioner the ease of allocating resources conveniently. As an example, systemic failures can be avoided by handling dependencies on shared databases early using data decoupling and eventually consistent data processing. On the same note, the development of DevOps tooling and automated test tools will eliminate risks associated with the process. The interdependency of issues related to technical, organisational, and process considerations makes the analysis distinct from the previous studies that usually consider the isolated factors [2].

5.1 Practice Implications

5.1.1 Technical

Move towards domain-driven design and use such service boundary and use saga patterns to provide distributed transactions [12]. API gateways and zero-trust security frameworks can overcome vulnerabilities and governance issues [16].

5.1.2 Process

Adopt mature DevOps practices (e.g. containerization (e.g., Docker), infrastructure-as-code (e.g., Terraform), and observability tools (e.g., Prometheus)) [5].

5.1.3 Organisational

Provide microservices training, establish cross-functional teams, and embed change management programs to enhance cooperation and autonomy [4].

5.2 Research Implications

The current paper offers a systematic risk-oriented analysis for microservices to the field of literature, which complements the currently available pattern-focused frameworks [13]. Future studies can employ the qualitative risk matrix to evaluate migration risks in other settings and environments because of the replicability of the model. The prominence of the process-related risks raises an understudied topic, which therefore requires the exploration of DevOps maturity and observability [3]. A different future research topic can also include quantitative risk assessment models in addition to a qualitative model.

5.3 Limitations

The nature of the instrument used in this research is secondary data which does not give much space to capture real life stories on migration. Although the migration risks are categorised, based on technical, organisational, and process, there is no systemic risk propagation between these categories, and all of these are not being modelled as interdependent. Risk scoring is performed using the general Impact x Likelihood approach on a 1-3 scale, available in literature and on the basis of expert evaluation, without direct interviews to the stakeholders. Mitigation measures are literature based but not ranked or cost analysed and proven empirically. The risk prioritisation model proposed does not validate at the industrial case studies, and the real impact of the proposed model on the real-life scenarios of SOA-to-MSA migrations remains at the latter to demonstrate.

6 Conclusion

The cross-technical, organisational, and process risk-oriented study of challenges in the migration from SOA to Microservices Architecture presents 20 key risks that can block the migration process. Such analyst risks as the reliance on shared databases and the absence of DevOps maturity (score = 9) highlight the importance of initiating different countermeasures aimed at considering architectural, operational and human issues. The suggested risk prioritisation matrix and mitigation strategies are pragmatic instruments that should be utilised by architects and technology leaders who plan reliable migrations.

The results indicate the relevance of operational preparedness and organizational integration, as well as technical refactoring. Considering the problem of high-impact risks and action items addressing them, the organisations will better succeed in the migration of their SOA to MSA environments and enjoy the benefits of microservices. The validation of the findings by empirical studies through industrial case studies, quantitative models of risk assessment and automated tools of risk assessment and migration planning should be the research area in the future. This will advance the model further and enable organisations to go through the maze of architectural modernisation.

References

1. R. Ram, P. Rodríguez, Software project risk management framework: A case study in cloud migration projects. J. Syst. Softw. **144**, 324-340 (2018). https://doi.org/10.1016/j.jss.2018.06.047
2. B.J. Fritzsch, J. Bogner, S. Wagner, A systematic literature review of migration strategies for microservices. Empir. Softw. Eng. **25**, (2020). https://doi.org/10.1007/s10664-020-09910-y
3. C.S. Li, C. Zhong, C. Zhang, Quality attributes of microservices architecture: Systematic literature review. J. Syst. Softw. **177**, (2021). https://doi.org/10.1016/j.jss.2020.1993
4. D.A. Razzaq, S.A.K. Ghayyur, Systematic mapping study: Monolithic to microservice awareness and challenges. Comput. Appl. Eng. Educ. **31**, (2023). https://doi.org/10.1002/cae.22586
5. E.M. Söylemez, B. Tekinerdogan, A.K. Tarhan, Challenges and solutions in microservices: A systematic review. Appl. Sci. **12**, (2022). https://doi.org/10.3390/app12115507
6. Y. Wang, H. Kadiyala, J. Rubin, Microservices architecture: promises and challenges. Empir. Softw. Eng. **26**, 63 (2021).
7. G.S.S. de Toledo, A. Martini, D.I. Knauss, Identifying architectural technical debt, principal, and interest in microservices: A multiple-case study. J. Syst. Softw. **177**, 110968 (2021). https://doi.org/10.1016/j.jss.2021.110968
8. M. Mazzara, N. Dragoni, A. Bucchiarone, A. Giaretta, S.T. Larsen, S. Dustdar, Microservices: migration of a mission critical system. IEEE Trans. Serv. Comput. **14**, 1464-1477 (2021). https://doi.org/10.1109/TSC.2018.2889087
9. A.S. Newman, Building microservices. (O'Reilly Media, 2015).
10. A.P. Di Francesco, P. Lago, Migrating towards microservice architectures: An industrial survey, in Proceedings of the IEEE International Conference on Software Architecture (ICSA), Seattle, WA, USA, July 23 (2018), 29-2909
11. B.J. Fritzsch, J. Bogner, S. Wagner, Microservices migration in industry: Intentions, strategies, and challenges, in Proceedings of the IEEE International Conference on Software Architecture (ICSA), Hamburg, Germany, December 05 (2019), 51-60
12. C.F. Ponce, G. Márquez, H. Astudillo, Migrating from monolithic architecture to microservices: A rapid review, in Proceedings of the IEEE Latin American Conference on Software Engineering, Concepcion, Chile, January 23 (2019), 1-7
13. D.V. Raj, R. Sadam, Patterns for migration of SOA-based applications to microservices architecture Journal of Web Engineering, River Publishers, July (2021), 1291-1307
14. E.V. Raj, K.S. Reddy, Best Practices and Strategy for the Migration of Service-Oriented Architecture-Based Applications to Microservices Architecture, In Advances in Distributed Computing and Machine Learning, Springer, Singapore, February 22 (2022), 439-449
15. F.V. Velepucha, P. Flores, A Survey on Microservices architecture: Principles, patterns and migration challenges. IEEE Access **11**, 88339-88358 (2023)
16. G.T. Yarygina, A.H. Bagge, Overcoming security challenges in microservice architectures, in Proceedings of the IEEE Symposium on Service-Oriented System Engineering (SOSE), Bamberg, Germany, May 17 (2018), 11-20
17. H.B. Kitchenham, S. Charters, Guidelines for performing systematic literature reviews in software engineering. Keele Univ. Tech. Rep, (2007)
18. Allied Market Research, Microservices architecture market size and forecast, 2020-2026. Allied Market Research, (2025). https://www.alliedmarketresearch.com/microservices-architecture-market
19. V.R. Raj, K.R. Bhukya, Assessing the impact of migration from SOA to microservices architecture. SN Comput. Sci. **4**, 577 (2023). https://doi.org/10.1007/s42979-023-01971-2
20. R. Nogueira, M. Sousa, H. Ferreira, An empirical study on motivations, activities, and challenges in migrating from monolithic systems to microservices. arXiv preprint arXiv:2408.10434, (2024). https://arxiv.org/abs/2408.10434