Enhancing the Successful Microservices Implementation

A Systematic Review Study

Dinda Ayu Hapsari[1], Teguh Raharjo[2], Anita Nur Fitriani[3], Bob Hardian[4]
Faculty of Computer Science, Universitas Indonesia, Jakarta, Indonesia[1, 2, 3, 4]
Computer Science Department-Binus Online Learning, Bina Nusantara University, Jakarta, Indonesia[2]

***Abstract*—Despite the widespread adoption of microservices across major global companies, a knowledge gap persists between best practices and real-world implementation challenges faced by practitioners. While existing literature provides extensive coverage of microservices patterns, limited evidence exists regarding how the practices are actually implemented in various organizational contexts. Thus, this study aims to address this gap through a systematic synthesis of empirically-validated microservices practices from recent peer-reviewed literature. We conducted a comprehensive systematic literature review and ultimately included thirty-four high-quality articles published between 2021 and 2025. We extracted 114 microservice practices and classified them into eight domains, which are Architecture and Design, Communication and Integration, Development and Deployment, Monitoring and Observability, Testing and Quality Assurance, Migration and Legacy Modernization, Security and Access Control, and Team Organization and Development Process. Architecture and Design practices and Team Organization and Process collectively account for nearly half of all identified practices, while Security and Access Control emerged as a significant research gap, with only 5.9 per cent of studies addressing this domain. To the best of our knowledge, this is the first systematic literature review that comprehensively synthesizes microservice implementation practices across multiple domains with explicit empirical validation in real-world contexts as inclusion criteria. This study provides a comprehensive catalogue of empirically-validated practices, offering structured guidance for practitioners and a foundation for future development of microservice implementation guidelines, contributing to more successful microservice projects and mitigated implementation risks.**

***Keywords*—Microservice; practice; systematic literature review; Kitchenham; PRISMA 2020**

I. Introduction

In the early 2010s, Netflix adopted an architecture that allowed small, autonomous teams to build and run specific services [1]. This success story marked an architectural turning point. This approach, called microservices, has since gained widespread recognition among practitioners and scholars [2]. Numerous organizations have successfully adopted microservices. Tech giants, including Amazon, Spotify, Uber, LinkedIn, and Twitter, have all implemented this architecture [3]. In the financial industry, microservice solutions can increase organizational agility [4]. Microservices are often integrated with agile development techniques [5]. Organizations benefit from modularity through loose coupling and high cohesion, distributed architecture and decentralization, scalable infrastructure, and reusable components.

However, despite microservices being well-established since 2014 and extensive literature on best practices, practitioners continue to face implementation challenges [2], [6]. Recent discoveries from the banking industry reveal anti-patterns in microservice implementations, as shown in Fig. 1. First, services become oversized and non-modular, contradicting core microservice principles. Large-scale microservice systems are likely to be expanded, altered, relocated, or deleted, leading to increased system complexity and potential technical debt and regression problems [7]. Second, the foundation code employs static global variables, creating service dependencies. If there is a problem with the foundation code, the microservices will also be impacted. Additionally, banking microservice implementations experience performance degradation due to database schemas that are inadequately designed to handle growing data volumes, leading to system performance bottlenecks. This fact is supported by the survey conducted by IBM Market Development & Insights. The survey found that 51% of data management is still geared toward monolithic app development. While such industry-specific findings provide valuable insights about deployment issues, they provide isolated information within specific organizational contexts and cannot capture broader practices.

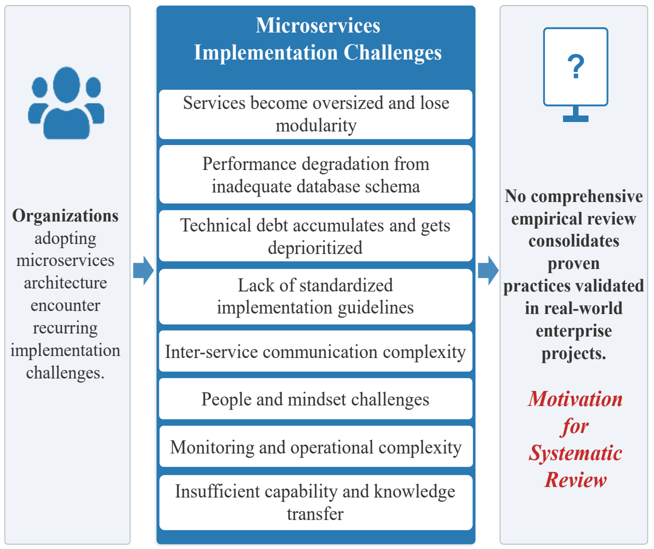
Organizations adopting microservices architecture encounter recurring implementation challenges.
Microservices Implementation Challenges
Services become oversized and lose modularity
Performance degradation from inadequate database schema
Technical debt accumulates and gets deprioritized
Lack of standardized implementation guidelines
Inter-service communication complexity
People and mindset challenges
Monitoring and operational complexity
Insufficient capability and knowledge transfer
?
No comprehensive empirical review consolidates proven practices validated in real-world enterprise projects.
Motivation for Systematic Review

Fig. 1. Research gaps: Microservice implementation practices.

Several empirical studies conducted through case studies and interviews have focused on fragmented areas of microservices implementation, making it difficult for practitioners to obtain comprehensive information about end-to-end microservice implementation. Furthermore, while prior secondary studies have mapped microservices architecture from various perspectives, these reviews either predate the recent wave of empirical research (2021-2025), focus predominantly on architectural aspects without covering cross-cutting domains such as security and team organization, or do not distinguish between theoretically proposed and empirically validated practices. This gap – the absence of a systematic synthesis of empirically grounded implementation practices across multiple domains – motivates the present study.

We carried out a comprehensive review of previous studies, following the protocols established by Kitchenham and Charters [8] and the PRISMA 2020 guidelines for reporting systematic reviews [9]. From an initial pool of 1,823 records, 34 empirically validated studies were selected for review. The main contribution of this study is a structured catalogue of 114 microservices implementation practices classified across eight domains, all grounded in empirical evidence from industrial and real-world contexts. These findings serve as evidence-based building blocks for developing microservice implementation guidelines and provide a foundation for future research on practice prioritization and context-specific adoption strategies.

We organize this study into six sections: Section II provides the general setting of microservices. Section III presents the related work. Section IV describes a detailed account of the research methodology. Section V reports and discusses the outcomes. Lastly, Section VI summarizes this study with threats to validity, contributions, and future work.

II. Background

Microservices architecture is an approach for developing a single application as a suite of small, independently deployable services, each running on its own process and communicating through lightweight mechanisms [10]. Unlike monolithic systems, where all components are tightly coupled within a single unit, microservices enable independent development, deployment, and scaling of individual services [11]. The architecture has been widely adopted by major technology companies and financial institutions seeking improved agility, scalability, and fault isolation [1], [3], [4].

However, the shift from monolithic to microservices architecture introduces significant challenges in areas such as service granularity, distributed data management, inter-service communication, and organizational coupling [12], [13]. Service granularity – determining the appropriate size and scope of each microservice – remains one of the most debated design decisions, as both overly fine-grained and overly coarse-grained services can lead to performance and maintenance issues [13]. Distributed data management across service-specific databases introduces challenges related to data consistency, transaction coordination, and cross-service querying [14]. Additionally, the organizational structure must align with the service architecture to avoid communication overhead, a principle formalized as Conway's Law [15].

Design patterns play a fundamental role in addressing these challenges. Waseem et al. [14] explored the degree to which practitioners adopt patterns for microservice data management. Cerny et al. [16] conceptualized a design pattern as numerous tested solutions to recurring design complications. Such design patterns represent fundamental practices in microservice implementation. Practices in microservice implementation encompass processes, techniques, platforms, reference architecture, algorithms, and quality assurance practices [17].

Although microservice research has expanded significantly in recent years, existing empirical studies have focused on fragmented areas [14], [18], making it difficult for practitioners to obtain a comprehensive view of implementation practices. No systematic literature review has specifically catalogued empirically validated microservices implementation practices across multiple domains. The following section presents a comparison with prior secondary studies and situates this study within the existing body of knowledge. This study aims to fill this gap by systematically identifying and categorizing microservice implementation practices from empirical literature.

III. Related Work

Several secondary studies have examined microservices from different perspectives. Pahl and Jamshidi [19] conducted one of the earliest systematic mapping studies (SMS) on microservices, classifying 21 primary studies published between 2014 and 2015 to characterize research directions in the field. Their study established an initial classification framework but was limited to a narrow timeframe and a small corpus. Alshuqayran et al. [20] extended this effort by mapping 33 studies focused on architectural challenges, quality attributes, and common architectural views in microservice systems. While both studies provided foundational overviews, they predated the substantial growth in empirical microservices research that occurred from 2018 onward.

Di Francesco et al. [21] performed a more comprehensive systematic mapping of 103 primary studies, investigating the focus of research on architecting with microservices and evaluating the potential for industrial adoption. Their study revealed that significant microservice systems must consist of larger numbers of services than covered by discovered studies at that time and highlighted a gap between academic proposals and industrial practice. Fritzsch et al. [17] conducted a rapid review on adopting microservices and DevOps in the cyber-physical systems (CPS) domain, identifying key challenges and proposing recommendations based on a supplementary case study. Söylemez et al. [22] conducted an SLR covering 85 primary studies published since 2014, identifying nine categories of challenges and 40 sub-categories with corresponding solution directions. However, their focus was on challenges and solutions rather than on cataloguing empirically validated implementation practices.

While these prior studies have contributed significantly to understanding microservices architecture, they share several common limitations that motivate the present work. First, most existing reviews accept both theoretical proposals and empirical studies without distinguishing between them, potentially mixing unvalidated concepts with evidence-based practices. Second, earlier reviews predominantly focus on architectural aspects, with limited coverage of cross-cutting domains such as team organization, security, and monitoring. Third, the majority cover literature published before 2021, missing the substantial body of empirical work produced in the recent period of microservices maturation. This study addresses these limitations by: 1) requiring empirical validation in industrial or real-world contexts as an explicit inclusion criterion, 2) covering literature from 2021 to 2025, capturing the most recent empirical evidence, and 3) organizing practices across eight implementation domains, providing a broader view than architecture-focused reviews alone. Table I summarizes the key differences between this study and prior secondary studies.

TABLE I. COMPARISON WITH PRIOR SECONDARY STUDIES

Author	Year	Type	Scope	Time Range	Empirical Validation
Pahl and Jamshidi [19]	2016	SMS	Classification framework	2014-2015	No
Alshuqayran et al. [20]	2016	SMS	Architectural challenges, quality attributes, and common architectural views	2014-2016	No
Di Francesco et al. [21]	2019	SMS	Architecting with microservices	Up to 2017	No
Söylemez et al. [22]	2022	SLR	Microservice challenge and solutions	2014-2022	No
Fritzsch et al. [17]	2023	Rapid review	Microservice and DevOps in CPS	2015-2021	Partial
This study	2026	SLR	Practices across domains	2021-2025	Yes

IV. SYSTEMATIC LITERATURE REVIEW

An SLR is a structured approach for locating, analyzing, and synthesizing all relevant studies addressing a particular research question or topic [23]. SLR consists of three processes: planning, conducting, and reporting [24]. We follow the instructions given by Kitchenham and Charters, which comprised the following main elements: 1) developing research question, 2) selecting a search strategy, 3) prescribing criteria for inclusion and exclusion, 4) choosing studies, 5) examining the quality of the included studies, and 6) extracting and synthesizing data [25]. While for reporting the systematic reviews, we follow PRISMA 2020.

A. Developing Research Questions

The study seeks to systematically identify microservice implementation practices reported in the available literature. In order to fulfill this goal, the research questions (RQ) guide the study:

RQ1: What microservices implementation practices are reported in empirical literature?

RQ2: How are microservices practices distributed across domains, and what patterns emerge?

RQ3: What gaps exist in current microservices implementation research?

The search strategy, inclusion and exclusion criteria are then created based on these RQs.

B. Defining a Search Strategy

The search strategy for the review encompasses the scope, search method, and search string. The scope consists of publication period and publication venue. The review includes literature published within the timeframe of 2021 to 2025. The databases selected for the search of scientific studies were Institute of Electrical and Electronics Engineers Inc. or IEEE Xplore, ProQuest, Sage Journals, ScienceDirect, Scopus, Springer Nature Link, Taylor & Francis, and Wiley Online Library. The choice of the databases reduces bias toward any single publisher's editorial preferences also minimizes the risk of missing relevant articles. All databases provide peer-reviewed article, making sure the quality.

We conducted an automated literature search for this research. Automated searching involves querying electronic databases using predefined search terms. Our first attempt was to perform a very specific search for answering the research questions. The initial keywords were practice, implementation, and microservice. However, the expected articles were not successfully retrieved. We decided to do another round of search using more general keywords: microservice, practice, project management, software project, and development process. To get the most of it, we also added method, case study, pattern, and approach. The following general search string was set: (microservice OR "micro-service") AND ("project management" OR "software project" OR "development process" OR team OR enterprise OR organization) AND (practice OR method OR "case study" OR implementation OR pattern OR approach).

To optimize the literature search process, this search string was modified iteratively for each database using Boolean operators. Each database required slightly different adaptations: IEEE Xplore utilized "All Metadata" fields, ScienceDirect and Scopus used the search scope within title-abstract-keywords fields, Wiley applied an "anywhere" field search approach, while Taylor & Francis and Springer Nature Link used the most simplified Boolean combination. Table II lists the final search string for each database, while Fig. 2 shows the distribution of articles per database in the initial search. The database search was conducted on 7 January 2026.

The search string was iteratively refined through pilot searches to balance recall and precision, adjusting terms based on initial result relevance. To validate the effectiveness of the search string, a set of known relevant papers identified during preliminary literature exploration, Bogner et al. [26], Ünlü et al. [27], Ait Said et al. [28] were used as seed papers. All three studies were confirmed to be retrievable by the final search string across the respective databases, providing confidence in the search string's ability to capture relevant empirical studies on microservices implementation practices.

TABLE II. FINAL SEARCH STRING

Database	Boolean Search
IEEE Explore	("All Metadata":microservice OR "All Metadata":"micro-service") AND ("All Metadata":"project management" OR "All Metadata":"software project" OR "All Metadata":"development process" OR "All Metadata":"team") AND ("All Metadata":practice OR "All Metadata":method OR "All Metadata":"case study")
ProQuest	(TI(microservice OR "micro-service") OR AB(microservice OR "micro-service") OR SU(microservice OR "micro-service")) AND (TI("project management" OR "software project" OR "development process" OR team) OR AB("project management" OR "software project" OR "development process" OR team) OR SU("project management" OR "software project" OR "development process" OR team)) AND (TI(practice OR method OR "case study") OR AB(practice OR method OR "case study") OR SU(practice OR method OR "case study"))
Sage Journals	(microservice OR "micro-service") AND ("project management" OR "software project" OR "development process" OR "team") AND (practice OR method OR "case study")
ScienceDirect	Title, abstract, keywords: (microservice OR "micro-service") AND ("project management" OR "software project" OR "development process" OR "team") AND (practice OR method OR "case study")
Scopus	TITLE-ABS-KEY ((microservice OR "micro-service") AND ("project management" OR "software project" OR "development process" OR "team") AND (practice OR method OR "case study"))
Springer Nature Link	(microservice OR "micro-service") AND ("project management" OR "software project" OR "development process" OR "team") AND (practice OR method OR "case study")
Taylor & Francis online	[[All: microservice] OR [All: "micro-service"]] AND [[All: "project management"] OR [All: "software project"] OR [All: "development process"] OR [All: "team"]] AND [[All: practice] OR [All: method] OR [All: "case study"]]
Wiley Online Library	"(microservice OR "micro-service")" anywhere and "(“project management" OR "software project" OR "development process" OR "team")" anywhere and "(practice OR method OR "case study")" anywhere

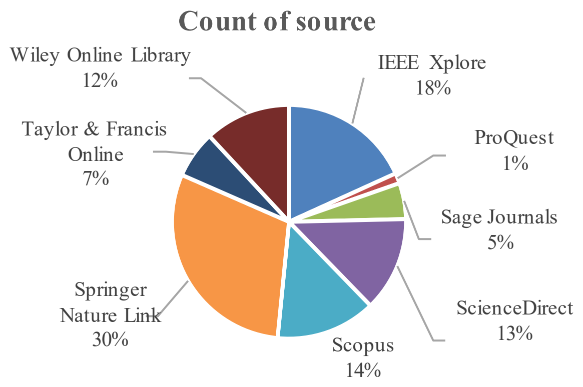

Fig. 2. Articles per source.

The choice of the databases ensures comprehensive coverage across technical (IEEE Xplore), business (Taylor & Francis online), high-impact research (ScienceDirect), international scope (Scopus), and emerging research (ProQuest). It is essential for capturing all aspects of microservice implementation practices in enterprise settings. A sum of 1823 papers were located using automated search with Springer Nature Link provided the most results ($n = 547$) while ProQuest the least amount (n=26).

C. Establishing Criteria for Inclusion and Exclusion

To ensure a relevant and focused selection of literature, inclusion and exclusion criteria were established and applied to the automatic search results. These criteria were defined to identify studies aligned with our research objectives about microservice practices, as detailed in Table III. The criteria were applied sequentially, starting from objective filters (IC.1-IC.4) before proceeding to content-based evaluation (IC.5-IC.7).

D. Selecting Studies

The initial search retrieved 1,823 records across 8 electronic databases. The duplicate records were then removed (EC.1), prompting 1706 records to go to the screening. Inclusion and exclusion criteria from Table III were applied in this phase. Only records published within the 2021-2025 timeframe (IC.2), English-language (IC.3), journal articles and conference papers (IC.4), and articles that focus on microservice implementations (IC.5) were retained, resulting in 224 going to the next step. Using IC.6 and EC.6, only 139 reports were retrieved.

TABLE III. STUDY SELECTION GUIDELINES

Inclusion Criteria (IC)	
IC.1	Not a duplicate publication
IC.2	Published between 2021 and 2025
IC.3	Written in English
IC.4	Peer-reviewed publications (journal articles, conference papers)
IC.5	Primary focus on microservices implementation practices, patterns, or approaches
IC.6	Full text accessible through institutional or open access
IC.7	Papers must report on the practical application, evaluation, or observed results of microservices implementation practices within a specific industrial or real-world context
Exclusion Criteria (EC)	
EC.1	Duplicate publications (retain most complete or recent version)
EC.2	Published outside 2021-2025 timeframe, or erratum, corrigendum, retraction, or correction
EC.3	Not written in English
EC.4	Grey literature, books, book chapters, theses, dissertations, technical reports, posters, workshops, editorials, non-research content, short papers (<4 pages), or not peer-reviewed papers
EC.5	Papers where microservices is not mentioned, or mentioned only tangentially (not primary focus)
EC.6	Full text not accessible through institutional or open access
EC.7	Papers that are purely theoretical, conceptual, or solution proposals (new models, frameworks, or methodologies) that lack documented application or validation in real-world projects

Lastly, we performed a full-text screening of the studies, implementing IC.7 and EC.7 as the gatekeeper, which generated thirty-six studies for comprehensive analysis. This review focused on empirical studies examining microservice implementations practices, development processes, and approaches in enterprise environments or real-world projects.

We excluded secondary studies and purely theoretical, conceptual, or solution proposals that lacked empirical application or validation in real-world projects to ensure focus on practical contributions.

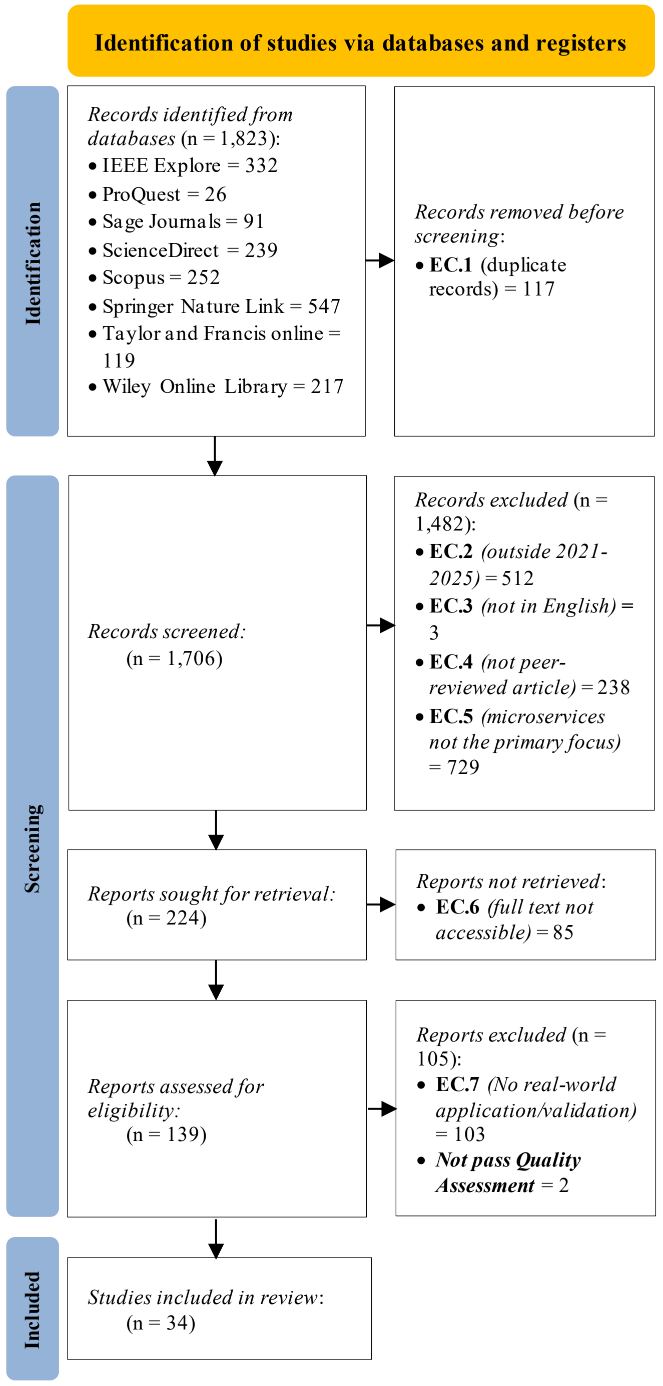
Identification of studies via databases and registers
Identification
Records identified from databases (n = 1,823):
• IEEE Explore = 332
• ProQuest = 26
• Sage Journals = 91
• ScienceDirect = 239
• Scopus = 252
• Springer Nature Link = 547
• Taylor and Francis online = 119
• Wiley Online Library = 217
Records removed before screening:
• EC.1 (duplicate records) = 117
Screening
Records screened: (n = 1,706)
Records excluded (n = 1,482):
• EC.2 (outside 2021-2025) = 512
• EC.3 (not in English) = 3
• EC.4 (not peer-reviewed article) = 238
• EC.5 (microservices not the primary focus) = 729
Reports sought for retrieval: (n = 224)
Reports not retrieved:
• EC.6 (full text not accessible) = 85
Reports assessed for eligibility: (n = 139)
Reports excluded (n = 105):
• EC.7 (No real-world application/validation) = 103
• Not pass Quality Assessment = 2
Included
Studies included in review: (n = 34)

Fig. 3. PRISMA 2020 flow diagram.

Fig. 3 shows the PRISMA 2020 flow diagram. Flow diagram allows the reader to quickly and easily understand the step-by-step techniques of a review and observe the systematic exclusion of irrelevant records throughout [29]. The reporting of the SLR was guided by the standards of the PRISMA Statement [30]. Complete reporting of PRISMA 2020 allows audiences to assess the methodological suitability, and therefore the credibility of the research outcomes [9].

E. Assessing the Quality of the Studies

Quality assessments (QA) were implemented to the 36 studies to maintain methodological rigor and enhance credibility. The QA process involved answering questions and scoring to improve decision-making quality. The QA questions were:

- Are the research objective and methodology clearly described?
- Are the findings clearly presented and supported by empirical evidence?
- Is the study relevant to enterprise microservices implementation practices?

The cutoff score was set to 2.0. Of 36 studies, 34 papers met stringent quality criteria and were accepted, while 2 were rejected. The systematic filtering and quality assessment procedures ultimately earned 34 peer-reviewed articles. The corpuses become the basis for our analysis, documented in Table V.

F. Extracting and Synthesizing Data

Data extraction was performed on the 34 papers that passed the selection and QA stage. To ensure consistency and traceability, a structured data extraction form was developed and managed using Parsifal. The extraction form was designed to capture key information of each included study. For each paper, the following information was extracted: paper type, research method, the primary and secondary practices domain covered in the paper, and the validation used to evaluate the reported findings.

Following extraction, the identified practices were classified into implementation domains through an iterative qualitative grouping process. During data extraction, the first author assigned each practice to a preliminary category based on its functional characteristics and the thematic context in which it appeared in the primary study. For instance, practices related to service decomposition, bounded context identification, and architectural pattern selection were grouped under Architecture and Design, while practices concerning Continuous Integration / Continuous Deployment (CI/CD) pipelines, containerization, and deployment automation were grouped under Development and Deployment. The resulting domain structure was reviewed by the co-authors to verify internal consistency, mutual exclusivity between domains, and completeness of coverage.

V. Results and Discussion

Following the SLR methodology outlined in Section IV, this section presents the comprehensive analysis of microservices implementation practices extracted from the studies. The results are aligned to address our research questions RQ1, RQ2, and RQ3. The results are presented in four main components: statistics, overview of selected studies, domain mapping summary, paper-domain matrix, and research gaps analysis. These components collectively address the research questions from different analytical perspectives.

A. Statistics

The final selection yielded 34 empirically validated studies published between 2021 and 2025. The temporal distribution shows 7 studies from 2021, 6 from 2022, 2 from 2023, 12 from 2024, and 7 from 2025, with the highest concentration in 2024, reflecting the growing maturity of microservice research. They comprised 23 journal articles (67.6%) and 11 conference papers (32.4%), indicating a predominantly peer-reviewed, high-quality corpus. The average quality assessment score across all studies was 2.84 out of 3.0, confirming the reliability of the evidence.

Table IV provides the distribution of publication venues, revealing the domination of three academic publishers: IEEE, Elsevier, and Springer. IEEE leads with 11 papers consisting of both conference proceedings and journals, including IEEE Transactions on Software Engineering, IEEE Transactions on Services Computing, and IT Professional, alongside major conference venues such as ICE-SEIP, ICSME, ASE, and ICPC. This reflects IEEE's strong presence in software engineering and systems research. Elsevier follows with 7 papers distributed across key journals, most notably the Journal of Systems and Software (3 publications) and Information and Software Technology (2 publications), confirming its prominence in empirical software engineering literature. Springer contributes 5 papers, primarily through Empirical Software Engineering (3 publications), alongside Journal of Cloud Computing and Service Oriented Computing and Applications. Wiley accounts for 4 papers, concentrated in Software: Practice and Experience (3 publications) and Software Testing, Verification, and Reliability. The remaining papers are distributed among ACM, EDP Sciences, Taylor & Francis, Springer Nature Switzerland, Science and Information Organization, and Scientific Publishing Institute, each contributing a single paper.

B. Overview of Selected Studies (RQ1)

In response to RQ1, the analysis of 34 selected studies yielded 114 microservices implementation practices, as presented in Table V. All practices were extracted from studies that demonstrated real-world validation, through case studies (17 papers), expert interviews (8 papers), experiments (5 papers), or practitioner surveys (4 studies), which are consistent with our inclusion criteria. This ensures the compiled practices reflect evidence-based implementation experience rather than purely theoretical recommendations. This can serve as the foundation for developing implementation guidelines for the enterprise.

TABLE IV. PUBLICATION VENUES

Publisher	No. of Papers
IEEE	11
Elsevier	7
Springer	5
Wiley	4
ACM	2
Others	5
Total	**34**

TABLE V. PRACTICES PER PAPER

Study ID	Author	Year	Practices Identified
S1	[31]	2021	• Automated security discussion identification using ML/DL • Security knowledge extraction from developer discussions • Identification and documentation of microservices-specific security challenges
S2	[32]	2021	• System-wide continuous integration pipeline for microservice quality assurance • Comprehensive quality assurance (static analysis, runtime testing, monitoring) • DevOps automation with automatic checks, • Small team ownership of microservices • Domain-Driven Design (DDD)-based bounded context identification for service decomposition • Agile workflow
S3	[33]	2021	• Build event-driven graphs for root cause analysis • Multi-source observability data integration • Embed the R&D efforts in the live environment with SRE experts and users
S4	[26]	2021	• Establish and enforce architectural guidelines, standards, and patterns • Specialized test automation for evolvability • Source code quality tools and metrics for microservices • Conscious technical debt management process
S5	[34]	2021	• Leverage open-source software (OSS) for migrating from monolith to distributed microservice architecture
S6	[35]	2021	• Architecture smell detection and refactoring for microservices • TOSCA-based architecture modelling for microservices
S7	[36]	2021	• Extracting microservices from monoliths based on the relationship-type • Service decomposition analysis
S8	[37]	2022	• Virtual team organization for DevOps and microservices • DevOps culture for rapid delivery and burden reduction
S9	[38]	2022	• Microservice as-a-service delivery model • API Gateway pattern • Cross-functional team integration for DevOps

Study ID	Author	Year	Practices Identified
			• Docker containerization for microservice deployment • DevOps and CI • Logging, Monitoring & Debugging
S10	[39]	2022	• Containerized microservice deployment for resource optimization • Dynamic resource scaling with container orchestration
S11	[40]	2022	• Real-time condition monitoring with microservice architecture • Cloud-edge collaborative microservice platform design • Deploy lightweight containerized microservices at edge
S12	[41]	2022	• Legacy system analysis and service identification • Service-oriented reengineering process (SPReaD) • DevOps pipeline for reengineered microservices • Support and feedback for DevOps monitoring
S13	[42]	2022	• Function point-based effort estimation for microservices • Event point-based size measurement for agile microservice projects • Code length-based software sizing for microservice-based agile projects
S14	[17]	2023	• DevOps and microservice method adaptation for cyber-physical systems • Architecture proposals for various application contexts • Organizational transformation for microservices and DevOps
S15	[43]	2023	• Deep learning-based anomaly detection on monitoring data • Continuous monitoring with feedback-driven model improvement • Cloud-based DL model deployment for anomaly detection
S16	[44]	2024	• Top management support as critical success factor • Adequate resource allocation for MSA projects • API Management & Standardization as tools and systems CSF • Project management as process CSF
S17	[45]	2024	• Performance monitoring and issue diagnosis • Association Call Identification • Heterogeneous Propagation Graph Construction • Root cause service localization
S18	[46]	2024	• Systematic Domain-Driven Design application process for microservices (DDD4M) • Bounded context determination for service boundaries • Context mapping for inter-service relationships • Domain expert involvement in microservice design
S19	[47]	2024	• Automated DORA metrics measurement from DevOps tooling • Tracking DORA metrics at individual microservice level and real-time • Data-driven continuous improvement using delivery metrics
S20	[48]	2024	• Apply Strangler Fig pattern for gradual migration • Database per service pattern for data isolation • Extensive monitoring for microservice environments • Unit and integration testing strategy for distributed services
S21	[49]	2024	• Maintain API backward compatibility • Implement API versioning • Cross-team collaboration for API evolution • Implement practices to prevent API design degradation
S22	[50]	2024	• Microservices vendor analysis • Reference architecture design using viewpoints • Domain analysis • API Gateway implementation • Asynchronous communication between microservice • CI/CD pipeline for deployment • Container-based deployment with orchestration • Authentication and authorization • Logging, Monitoring & Debugging
S23	[27]	2024	• Agile methodology compatibility with microservices • Familiar design techniques adapted for microservices • Adapted testing practices for microservice
S24	[51]	2024	• Multi-criteria microservice boundary determination • Apply Domain-Driven Design to identify bounded contexts for migration

Study ID	Author	Year	Practices Identified
			• DevOps team involvement in migration planning
S25	[52]	2024	• Domain model-driven design for microservice decomposition • Layered microservice architecture design • Integration of heterogeneous systems
S26	[53]	2024	• Modular monolith as intermediate migration step • Domain model refactoring • Performance evaluation at each migration step
S27	[54]	2024	• Technical debt identification and management in MSA • Fostering communication to mitigate inadvertent TD. • Organizational structure alignment with microservice boundaries
S28	[55]	2025	• Reusability Maturity Framework (RMF) for microservices • DDD and MDD integration for reusable service design • Standardized service templates for reuse
S29	[56]	2025	• Metadata-driven API design for service reusability • Architecture refactoring catalogue for microservices • Observability improvement through architecture refactoring
S30	[57]	2025	• Technology heterogeneity governance framework • Develop an inner-source or open-source process to foster collaboration between teams on cross-cutting tools and libraries. • Introducing a task force team as a front-running team • Establish an education program for the microservices teams • Documentation of standardization
S31	[58]	2025	• Consumer-driven contract testing for API compatibility • Contract testing integration into continuous integration pipeline • Establish formal API contract specifications between consumer and provider
S32	[59]	2025	• Automated API fuzz testing with search-based techniques • Integration of automated test generation into CI pipeline • White-box system testing for microservices
S33	[60]	2025	• CI/CD pipeline for reusable microservice assets • InnerSource practices for microservice contribution • Automated compliance checks for reusable assets
S34	[61]	2025	• Docker containerization for microservice deployment • Kubernetes orchestration for container management • Decompose monolithic enterprise systems into microservices

C. Domain Mapping Summary (RQ2)

Based on a systematic analysis of 34 selected studies, the microservice research landscape is wide. It covers complete yet complex implementation aspects. To address RQ2, the 114 identified practices were classified into eight distinct domains. The finding reveals emphasis on foundational architectural practices, team organization & development process, development & deployment, monitoring & observability, and testing & quality assurance. Table VI provides the domain mapping summary, showing the distribution of practices and the coverage of each domain across the studies, while Table VII presents practices per study-domain matrix.

TABLE VI. DOMAIN MAPPING SUMMARY

Domain	Total Practices	Studies	Distribution %	Coverage %
Architecture & Design	29	S1, S6, S7, S10. S15, S18, S19, S22, S23, S24, S25, S27, S28, S29, S30, S31, S32	25.4%	50.0%
Communication & Integration	9	S1, S5, S9, S10, S14, S26	7.9%	17.6%
Development & Deployment	16	S1, S9, S11, S13, S15, S22, S23, S28, S29, S33, S34	14.0%	32.4%
Monitoring & Observability	15	S1, S2, S6, S11, S13, S16, S21, S29, S31	13.2%	26.5%
Testing & Quality Assurance	10	S4, S6, S15, S18, S20, S26, S32	8.8%	20.6%
Migration & Legacy Modernization	8	S6, S7, S8, S13, S19, S20	7.0%	17.6%
Security & Access Control	4	S1, S3	3.5%	5.9%
Team Organization & Development	23	S5, S9, S12, S14, S15, S16, S17, S18, S19, S21, S24, S25, S28, S30, S34	20.2%	44.1%
TOTAL	**114**		**100.0%**	

TABLE VII. PAPER-DOMAIN MATRIX

Study ID	Practices Count per Domain							
	Architecture & Design	Communication & Integration	Development & Deployment	Monitoring & Observability	Testing & Quality Assurance	Migration & Legacy Modernization	Security & Access Control	Team Organization & Development
S1							3	
S2	1		2		1			2
S3				2				1
S4	3				1			
S5						1		
S6	2							
S7	1					1		
S8								2
S9		1	1					1
S10			2					
S11	1		1	1				
S12			1	1		2		
S13								3
S14	1		1					1
S15			1	2				
S16		1						3
S17				4				
S18	3							1
S19				2				1
S20	1			1	1	1		
S21		3						1
S22	3	2	2	1			1	
S23	1				1			1
S24	1					1		1
S25	2	1						
S26					1	2		
S27	1							2
S28	2		1					
S29	2			1				
S30	3							2
S31		1			2			
S32					3			
S33			2					1
S34	1		2					
TOTAL per Domain	**29**	**9**	**16**	**15**	**10**	**8**	**4**	**23**

Architecture & Design emerged as the most prominent domain, contributing 29 practices (25.4% of all practices) and drawn from 17 studies (50.0% coverage of all selected studies). This dominance reflects the foundational role of architectural decisions in adoption of microservices, including practices such as domain-driven decomposition [62], API gateway implementation, and service granularity management. The high coverage indicates the architectural concerns are consistently addressed across the literature, regardless of the application context.

Team Organization & Development Process ranked second with 23 practices (20.2%) from 15 studies (44.1% coverage of 34 studies). This finding underscores the sociotechnical nature of microservices, where organizational alignment, cross-functional team structures, and agile workflow practices are essential, complementary to architectural design. Practices in this domain include DevOps culture adoption, team autonomy, and knowledge sharing.

Development & Deployment domain contributed 16 practices (14.0%) from 11 studies (32.4% coverage of selected studies), emphasizing CI/CD pipelines, containerization, and infrastructure in the microservices ecosystem. Monitoring & Observability followed with 15 practices identified (13.2%) extracted from 9 studies (26.5% coverage), highlighting the critical need for tracing, logging, and anomaly detection in microservice environments. Testing & Quality Assurance (10 practices, 8.8%) and Migration & Legacy Modernization (8 practices, 7.0%) showed moderate representation, each covered by 6-7 studies. These domains address practical challenges such as contract testing, integration testing strategies, and gradual migration approaches from monolithic systems.

Notably, Security & Access Control was the least represented domain with only 4 practices (3.5%) from 2 studies (5.9% coverage of articles included). This significant gap suggests that security aspects of microservices implementation remain underexplored in empirical literature, despite being a critical concern in microservices architecture. This finding directly addresses RQ3 by identifying security as an area requiring further research attention.

The paper-domain matrix provides a detailed map of each study's contribution across the eight practice domains. Most studies (17 studies) contributed to two domains, while 8 studies focused on a single domain and 9 addressed three or more domains. The most comprehensive study was S22, contributing across five domains. Architecture & Design and Team Organization & Development Process were the most frequently coexisting domains, consistent with Conway's Law about the relationship between system architecture and organizational structure [15].

D. Research Gap Analysis (RQ3)

To address RQ3, the research gaps in current microservice implementation research were analyzed through two dimensions: domain coverage gaps, as shown in Table VI and temporal gaps in Table VIII. The coverage percentage, counted as the proportion of the studies addressing each domain out of the total 34 studies, reveals a significant discrepancy across domain. Security & Access Control represents the most critical gap, with only 5.9% coverage (2 studies) addressing this domain. Migration & Legacy Modernization and Communication & Integration (17.6% coverage each) remain relatively underexplored despite being common challenges.

To identify whether research gaps persist or are being addressed over time, a temporal analysis was conducted by mapping practices across publication years. The temporal distribution reveals several noteworthy patterns. Four domains - Architecture & Design, Development & Deployment, Monitoring & Observability, and Team Organization & Development Process - have remained active throughout all five years, showing sustained research interest. However, other domains exhibit significant temporal discontinuities. Security & Access Control shows the most concerning temporal pattern, with contributions appearing only in 2021 (3 practices) and 2024 (1 practice), and a complete absence in 2022, 2023, and 2025. Migration & Legacy Modernization was absent in both 2023 and 2025. This is particularly concerning given that many organizations are still in the process of transitioning from monolithic to microservices architectures. The Communication & Integration domain had no contributions in 2021 and 2023, while Testing & Quality Assurance was absent in 2022 and 2023. These suggest that research attention to these domains has been sporadic rather than sustained.

E. Cross-Domain Analysis and Synthesis

Beyond the domain-level distribution, the paper-domain matrix (Table VII) reveals notable cross-domain co-occurrence patterns that provide deeper insights into how microservice practices relate to one another. The most prominent pattern is the frequent co-occurrence of Architecture and Design practices with Team Organization and Development Process practices. Of the 34 selected studies, those that address architectural concerns such as decomposition and bounded context identification also tend to address team-related practices such as small team ownership and cross-functional collaboration. This observation aligns with Conway's Law [15], which posits that the structure of a system mirrors the communication structure of the organization that produces it. The empirical evidence from this review reinforces that architectural decisions in microservices are inseparable from organizational decisions, suggesting that organizations cannot effectively adopt microservices architecture without simultaneously restructuring their teams around service boundaries.

A second notable pattern involves the relationship between Development and Deployment practices and Monitoring and Observability. Studies that report CI/CD pipeline implementation frequently co-report monitoring practices such as automated anomaly detection, DORA metrics tracking, and distributed tracing. This suggests practical dependency; effective continuous deployment at scale presupposes robust monitoring capabilities to detect and respond to deployment-related issues in real time. Organizations planning to adopt CI/CD for microservices should therefore invest in monitoring infrastructure concurrently rather than sequentially.

Regarding evidence strength, the identified practices vary considerably in the breadth of their empirical support. Practices such as Domain-Driven Design for service decomposition, CI/CD pipeline automation, and containerization with Docker and Kubernetes appear across multiple independent studies and organizational contexts, suggesting robust evidence. In contrast, more specialized practices such as automated security discussion identification or heterogeneous propagation graph construction are reported in single studies, making their generalizability less certain. Practitioners should exercise caution when adopting practices with limited empirical coverage and consider piloting them within their specific organizational context.

The security gap identified in RQ3 warrants particular attention. Security and Access Control practices account for only 5.9% of all identified practices, and the temporal analysis shows that the security-focused studies appear only sporadically across the review period. This is concerning given the distributed nature of microservices, which inherently expands the attack surface through increased inter-service communication, multiple authentication points, and distributed data storage. This aligns with the implementation challenges described in the introduction, reinforcing the need for proactive security integration in microservices adoption. Organizations adopting microservices should proactively integrate security practices from the outset rather than treating security as a post-deployment concern.

TABLE VIII. TEMPORAL DISTRIBUTION OF PRACTICES PER DOMAIN

Domain	**Publication Year**					**Total**
	2021	**2022**	**2023**	**2024**	**2025**	
Architecture & Design	7	1	1	12	8	29
Communication & Integration	-	1	-	7	1	9
Development & Deployment	2	5	2	2	5	16
Monitoring & Observability	2	2	2	8	1	15
Testing & Quality Assurance	2	-	-	3	5	10
Migration & Legacy Modernization	2	2	-	4	-	8
Security & Access Control	3	-	-	1	-	4
Team Organization & Development	3	6	1	10	3	23
TOTAL	**21**	**17**	**6**	**47**	**23**	**114**

VI. CONCLUSION AND FUTURE WORK

This section summarizes the key findings, discusses the limitations of this study, highlights the contributions and implications for practitioners and researchers, and outlines directions for future research.

A. Conclusion

This systematic literature review presents a structured synthesis of empirical microservice implementation practices, extracted from 34 empirically validated studies published between 2021 and 2025. Through rigorous analysis, we have identified and categorized 114 implementation practices that reflect the current state of microservice adoption in real-world enterprise environments. The practices were categorized into eight domains: Architecture and Design, Communication and Integration, Development and Deployment, Monitoring and Observability, Testing and Quality Assurance, Migration and Legacy Modernization, Security and Access Control, and Team Organization and Development Process.

In response to RQ1, the study successfully catalogued a structured set of microservices practices that are grounded in real-world industrial experience. For RQ2, the analysis revealed that Architecture Design and Team Organization & Development Process collectively cover nearly half of all identified practices, confirming the importance of technical design and organizational alignment in microservices adoption, consistent with the principles of Conway's Law. For RQ3, the study highlighted Security & Access Control (5.9% coverage) as the most significant research gap. The explicit requirement for empirical validation as an inclusion criterion (IC7) distinguishes this review from prior secondary studies, ensuring that synthesized practices reflect validated industrial experience rather than theoretical speculation. The limitations described in Section IV B should be considered when interpreting the findings.

B. *Threats to Validity*

Several threats to validity of this study should be acknowledged. Regarding internal validity, the domain classification of 114 practices was conducted by the first author based on characteristics and thematic context and subsequently reviewed by the co-authors for consistency. However, formal inter-rater reliability measurement and independent parallel coding were not conducted, which may have introduced classification inconsistencies. Additionally, the quality assessment involved subjective judgment in scoring, using structured questions and a predefined cutoff threshold, which helped to maintain consistency.

Concerning external validity, the search was limited to eight electronic databases, and grey literature sources such as technical blogs, industry reports, and white papers were excluded. While this decision was made to ensure the methodological rigor associated with peer-reviewed literature, it may have excluded relevant practitioners' knowledge not yet captured in academic publications. The restriction to English-language publications may also have excluded relevant studies published in other languages. Furthermore, while the search string was validated against three known relevant seed papers, formal recall and precision metrics were not computed across the full corpus, which means that some relevant studies may not have been retrieved.

With respect to construct validity, the granularity of reported practices varies considerably across primary studies, ranging from high-level organizational factors (e.g., top management support) to specific technical techniques (e.g., heterogeneous propagation graph construction). This variation reflects the heterogeneous reporting conventions in the empirical literature rather than an analytical choice by the authors. As a result, the practices catalogue should be interpreted as a synthesis of reported evidence at varying levels of abstraction. Finally, regarding conclusion validity, publication bias in the primary studies may favor positive implementation outcomes, potentially underrepresenting failed or problematic practices.

C. *Practical and Academic Contributions*

This study offers significant contributions to both industry practices and academic research. For practitioners and enterprise architects, this research provides an evidence-based foundation for microservices implementation decisions. The identified practices provide a knowledge base that practitioners can use as a reference when planning and implementing microservices. The industry orientation of the reviewed literature ensures that identified practices are grounded in real-world experience and validated through practical deployment scenarios. This empirical foundation can increase the likelihood of microservice successful implementation. For example, organizations initiating microservices adoption can prioritize the Architecture and Design and Team Organization and Development Process domains as starting points, given their dominant coverage in the literature. Second, Monitoring and Observability infrastructure should be established before or concurrently with scaling microservices to production environments. The empirical evidence indicates that successful CI/CD implementations are typically accompanied by robust monitoring capabilities. Third, the identified gap in the Security and Access Control signals that practitioners should not rely solely on academic literature for microservices security guidance and may need to supplement with industry security frameworks and integrate security validation into their CI/CD pipelines. The domain-based categorization allows organizations to evaluate their own implementation readiness, highlighting areas of strength and identifying domains that demand additional attention. Context-specific practice selection is proposed as a direction for future work. From an academic perspective, this study advances microservices research by providing the implementation practices derived from systematic literature analysis. The eight-domain categorization facilitates future research on implementation guidelines.

D. *Future Work*

This study provides a solid foundation for future research. The practice catalogue can serve as the basis for developing a comprehensive microservice implementation guideline using Design Science Research (DSR) methodology. Such a guideline could also be structured around the eight practice domains, providing domain-specific recommendations. The DSR artifact would undergo evaluation through expert reviews and industrial case studies to ensure its practical applicability across different organizations. Additionally, future research should address the identified gaps by conducting dedicated empirical studies on security practices in microservice architecture, developing evidence-based migration strategies, and investigating communication patterns in large-scale deployments. Future research could apply the Analytical Hierarchy Process (AHP) to establish practice prioritization tailored to different organizational contexts.

Acknowledgment

During the preparation of this manuscript/study, the author(s) used Claude AI (Anthropic) for language editing, text refinement to enhance readability, and academic writing quality. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Authors' Contribution

Dinda Ayu Hapsari: Conceptualization, Methodology, Formal Analysis, Resources, Writing – original draft, Writing – review and editing, Visualization, Funding acquisition. Teguh Raharjo: Conceptualization, Validation, Formal Analysis, Writing – review and editing, Supervision. Anita Nur Fitriani: Writing – review and editing. Bob Hardian: Writing – review and editing. All authors have read and agreed to the published version of the manuscript.

Data Availability

The data that support the findings in this research are openly available in Zenodo at https://doi.org/10.5281/zenodo.19150851.

References

[1] C. Henríquez, J. D. R. Valencia, and G. Sánchez, “Architectural Evolution from Monolithic to Microservices in Scalable Systems: A Case Study of Netflix,” vol. 23, 2025.

[2] M. Washeem, D. T. Maini, S. Jhingran, G. Mishra, and P. K. Mishra, “A comprehensive analysis of Microservices,” vol. 20, no. 20, 2022.

[3] G. Blinowski, A. Ojdowska, and A. Przybylek, “Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation,” IEEE Access, vol. 10, pp. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[4] M. Mazzara, N. Dragoni, A. Bucchiarone, A. Giaretta, S. T. Larsen, and S. Dustdar, “Microservices: Migration of a Mission Critical System,” IEEE Trans. Serv. Comput., vol. 14, no. 5, pp. 1464–1477, Sep. 2021, doi: 10.1109/TSC.2018.2889087.

[5] A. Aggarwal and V. Singh, “Migration aspects from monolith to distributed systems using software code build and deployment time and latency perspective,” TELKOMNIKA, vol. 22, no. 4, p. 854, Aug. 2024, doi: 10.12928/telkomnika.v22i4.25655.

[6] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, “Challenges and solutions when adopting DevSecOps: A systematic review,” Information and Software Technology, vol. 141, p. 106700, Jan. 2022, doi: 10.1016/j.infsof.2021.106700.

[7] M. E. Gortney et al., “Visualizing Microservice Architecture in the Dynamic Perspective: A Systematic Mapping Study,” IEEE Access, vol. 10, pp. 119999–120012, 2022, doi: 10.1109/ACCESS.2022.3221130.

[8] B. Kitchenham and S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering.” 2007.

[9] M. J. Page et al., “The PRISMA 2020 statement: an updated guideline for reporting systematic reviews,” BMJ, p. n71, Mar. 2021, doi: 10.1136/bmj.n71.

[10] M. Fowler and J. Lewis, “Microservices.” Accessed: Mar. 20, 2026. [Online]. Available: https://martinfowler.com/articles/microservices.html

[11] V. Velepucha and P. Flores, “A Survey on Microservices Architecture: Principles, Patterns and Migration Challenges,” IEEE Access, vol. 11, pp. 88339–88358, 2023, doi: 10.1109/ACCESS.2023.3305687.

[12] A. Martínez Saucedo, G. Rodríguez, F. Gomes Rocha, and R. P. D. Santos, “Migration of monolithic systems to microservices: A systematic mapping study,” Information and Software Technology, vol. 177, p. 107590, Jan. 2025, doi: 10.1016/j.infsof.2024.107590.

[13] S. Hassan, R. Bahsoon, and R. Buyya, “Systematic scalability analysis for microservices granularity adaptation design decisions,” Softw Pract Exp, vol. 52, no. 6, pp. 1378–1401, Jun. 2022, doi: 10.1002/spe.3069.

[14] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, “Design, monitoring, and testing of microservices systems: The practitioners’ perspective,” Journal of Systems and Software, vol. 182, p. 111061, Dec. 2021, doi: 10.1016/j.jss.2021.111061.

[15] M. Fowler, “Conway’s Law,” martinfowler.com. Accessed: Feb. 23, 2026. [Online]. Available: https://martinfowler.com/bliki/ConwaysLaw.html

[16] T. Cerny, A. S. Abdelfattah, A. A. Maruf, A. Janes, and D. Taibi, “Catalog and detection techniques of microservice anti-patterns and bad smells: A tertiary study,” Journal of Systems and Software, vol. 206, p. 111829, Dec. 2023, doi: 10.1016/j.jss.2023.111829.

[17] J. Fritzsch et al., “Adopting microservices and DevOps in the cyber-physical systems domain: A rapid review and case study,” Softw Pract Exp, vol. 53, no. 3, pp. 790–810, Mar. 2023, doi: 10.1002/spe.3169.

[18] X. Zhou et al., “Revisiting the practices and pains of microservice architecture in reality: An industrial inquiry,” Journal of Systems and Software, vol. 195, p. 111521, Jan. 2023, doi: 10.1016/j.jss.2022.111521.

[19] C. Pahl and P. Jamshidi, “Microservices: A Systematic Mapping Study:,” in Proceedings of the 6th International Conference on Cloud Computing and Services Science, Rome, Italy: SCITEPRESS - Science and Technology Publications, 2016, pp. 137–146. doi: 10.5220/0005785501370146.

[20] N. Alshuqayran, N. Ali, and R. Evans, “A Systematic Mapping Study in Microservice Architecture,” in 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China: IEEE, Nov. 2016, pp. 44–51. doi: 10.1109/SOCA.2016.15.

[21] P. Di Francesco, P. Lago, and I. Malavolta, “Architecting with microservices: A systematic mapping study,” Journal of Systems and Software, vol. 150, pp. 77–97, Apr. 2019, doi: 10.1016/j.jss.2019.01.001.

[22] M. Söylemez, B. Tekinerdogan, and A. Kolukısa Tarhan, “Challenges and Solution Directions of Microservice Architectures: A Systematic Literature Review,” Applied Sciences, vol. 12, no. 11, p. 5507, May 2022, doi: 10.3390/app12115507.

[23] R. Van Dinter, B. Tekinerdogan, and C. Catal, “Automation of systematic literature reviews: A systematic literature review,” Information and Software Technology, vol. 136, p. 106589, Aug. 2021, doi: 10.1016/j.infsof.2021.106589.

[24] P. Marnada, T. Raharjo, B. Hardian, and A. Prasetyo, “Agile project management challenge in handling scope and change: A systematic literature review,” Procedia Computer Science, vol. 197, pp. 290–300, 2022, doi: 10.1016/j.procs.2021.12.143.

[25] P. Haindl, P. Kochberger, and M. Sveggen, “A Systematic Literature Review of Inter-Service Security Threats and Mitigation Strategies in Microservice Architectures,” IEEE Access, vol. 12, pp. 90252–90286, 2024, doi: 10.1109/ACCESS.2024.3406500.

[26] J. Bogner, J. Fritzsch, S. Wagner, and A. Zimmermann, “Industry practices and challenges for the evolvability assurance of microservices: An interview study and systematic grey literature review,” Empir Software Eng, vol. 26, no. 5, p. 104, Sep. 2021, doi: 10.1007/s10664-021-09999-9.

[27] H. Ünlü, D. E. Kennouche, G. K. Soylu, and O. Demirörs, “Microservice-based projects in agile world: A structured interview,” Information and Software Technology, vol. 165, p. 107334, Jan. 2024, doi: 10.1016/j.infsof.2023.107334.

[28] M. Ait Said, L. Belouaddane, S. Mihi, and A. Ezzati, “A Systematic Framework To Enhance Reusability In Microservice Architecture,” IJCDS, vol. 17, no. 1, pp. 1–18, Jan. 2025, doi: 10.12785/ijcds/1571009176.

[29] N. R. Haddaway, M. J. Page, C. C. Pritchard, and L. A. McGuinness, “PRISMA2020 : An R package and Shiny app for producing PRISMA 2020-compliant flow diagrams, with interactivity for optimised digital transparency and Open Synthesis,” Campbell Systematic Reviews, vol. 18, no. 2, p. e1230, Jun. 2022, doi: 10.1002/cl2.1230.

[30] R. Sarkis-Onofre, F. Catalá-López, E. Aromataris, and C. Lockwood, “How to properly use the PRISMA Statement,” Syst Rev, vol. 10, no. 1, pp. 117, s13643-021-01671-z, Dec. 2021, doi: 10.1186/s13643-021-01671-z.

[31] A. Rezaei Nasab et al., “Automated identification of security discussions in microservices systems: Industrial surveys and experiments,” Journal of Systems and Software, vol. 181, p. 111046, Nov. 2021, doi: 10.1016/j.jss.2021.111046.

[32] D. Pianini and A. Neri, “Breaking down monoliths with Microservices and DevOps: an industrial experience report,” in 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME), Luxembourg: IEEE, Sep. 2021, pp. 505–514. doi: 10.1109/ICSME52107.2021.00051.

[33] H. Wang et al., “Groot: An Event-graph-based Approach for Root Cause Analysis in Industrial Settings,” in 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), Melbourne, Australia: IEEE, Nov. 2021, pp. 419–429. doi: 10.1109/ASE51524.2021.9678708.

[34] A. Vera-Baquero, O. Phelan, P. Slowinski, and J. Hannon, “Open Source Software as the Main Driver for Evolving Software Systems Toward a

Distributed and Performant E-Commerce Platform: A Zalando Fashion Store Case Study," IT Prof., vol. 23, no. 1, pp. 34–41, Jan. 2021, doi: 10.1109/MITP.2020.2994993.

[35] J. Soldani, G. Muntoni, D. Neri, and A. Brogi, "The μ TOSCA toolchain: Mining, analyzing, and refactoring microservice-based architectures," Softw Pract Exp, vol. 51, no. 7, pp. 1591–1621, Jul. 2021, doi: 10.1002/spe.2974.

[36] L. J. Kirby, E. Boerstra, Z. J. C. Anderson, and J. Rubin, "Weighing the Evidence: On Relationship Types in Microservice Extraction," in 2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC), Madrid, Spain: IEEE, May 2021, pp. 358–368. doi: 10.1109/ICPC52881.2021.00041.

[37] X. Zhou, H. Huang, H. Zhang, X. Huang, D. Shao, and C. Zhong, "A Cross-Company Ethnographic Study on Software Teams for DevOps and Microservices: Organization, Benefits, and Issues," in 2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Pittsburgh, PA, USA: IEEE, May 2022, pp. 1–10. doi: 10.1109/ICSE-SEIP55303.2022.9794010.

[38] M. De Bayser et al., "DevOps and microservices in scientific system development: experience on a multi-year industry research project," in Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing, Virtual Event: ACM, Apr. 2022, pp. 1452–1455. doi: 10.1145/3477314.3507317.

[39] A. Mulahuwaish, S. Korbel, and B. Qolomany, "Improving datacenter utilization through containerized service-based architecture," J Cloud Comp, vol. 11, no. 1, p. 44, Sep. 2022, doi: 10.1186/s13677-022-00319-0.

[40] H. Yang, S. K. Ong, A. Y. C. Nee, G. Jiang, and X. Mei, "Microservices-based cloud-edge collaborative condition monitoring platform for smart manufacturing systems," International Journal of Production Research, vol. 60, no. 24, pp. 7492–7501, Dec. 2022, doi: 10.1080/00207543.2022.2098075.

[41] C. E. Da Silva, Y. D. L. Justino, and E. Adachi, "SPReaD: service-oriented process for reengineering and DevOps: Developing microservices for a Brazilian state department of taxation," SOCA, vol. 16, no. 1, pp. 1–16, Mar. 2022, doi: 10.1007/s11761-021-00329-x.

[42] H. Unlu, T. Hacaloglu, F. Buber, K. Berrak, O. Leblebici, and O. Demirors, "Utilization of Three Software Size Measures for Effort Estimation in Agile World: A Case Study," in 2022 48th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Gran Canaria, Spain: IEEE, Aug. 2022, pp. 239–246. doi: 10.1109/SEAA56994.2022.00045.

[43] A. Hrusto, E. Engström, and P. Runeson, "Towards optimization of anomaly detection in DevOps," Information and Software Technology, vol. 160, p. 107241, Aug. 2023, doi: 10.1016/j.infsof.2023.107241.

[44] M. G. Amri, T. Raharjo, A. N. Fitriani, and N. R. P. Hutasuhut, "Critical Success Factors of Microservices Architecture Implementation in the Information System Project," ijacsa, vol. 15, no. 10, 2024, doi: 10.14569/IJACSA.2024.0151064.

[45] L. Tao et al., "Diagnosing Performance Issues for Large-Scale Microservice Systems With Heterogeneous Graph," IEEE Trans. Serv. Comput., vol. 17, no. 5, pp. 2223–2235, Sep. 2024, doi: 10.1109/TSC.2024.3402172.

[46] C. Zhong et al., "Domain-Driven Design for Microservices: An Evidence-Based Investigation," IEEE Trans. Software Eng., vol. 50, no. 6, pp. 1425–1449, Jun. 2024, doi: 10.1109/TSE.2024.3385835.

[47] J. Rüegger, M. Kropp, S. Graf, and C. Anslow, "Fully Automated DORA Metrics Measurement for Continuous Improvement," in Proceedings of the 2024 International Conference on Software and Systems Processes, München Germany: ACM, Sep. 2024, pp. 36–45. doi: 10.1145/3666015.3666020.

[48] V. L. Nogueira, F. S. Felizardo, A. M. M. M. Amaral, W. K. G. Assunção, and T. E. Colanzi, "Insights on Microservice Architecture Through the Eyes of Industry Practitioners," in 2024 IEEE International Conference on Software Maintenance and Evolution (ICSME), Flagstaff, AZ, USA: IEEE, Oct. 2024, pp. 765–777. doi: 10.1109/ICSME58944.2024.00080.

[49] A. Lercher, J. Glock, C. Macho, and M. Pinzger, "Microservice API Evolution in Practice: A Study on Strategies and Challenges," Journal of Systems and Software, vol. 215, p. 112110, Sep. 2024, doi: 10.1016/j.jss.2024.112110.

[50] M. Söylemez, B. Tekinerdogan, and A. K. Tarhan, "Microservice reference architecture design: A multi-case study," Softw Pract Exp, vol. 54, no. 1, pp. 58–84, Jan. 2024, doi: 10.1002/spe.3241.

[51] A. TaeiZadeh, Z. Lotfi, and A. J. Ramadhan, "Microservices Boundary Determination Migration in DevOps: A Case Study," BIO Web Conf., vol. 97, p. 00122, 2024, doi: 10.1051/bioconf/20249700122.

[52] Y. Zhao, J. Liang, Y. Yang, Y. Zhang, and D. Han, "Power Grid Infrastructure Business System Architecture Based on Domain Model-Driven Design Method," in 2024 5th International Conference on Smart Grid and Energy Engineering (SGEE), Nanchang, China: IEEE, Nov. 2024, pp. 167–173. doi: 10.1109/SGEE64306.2024.10865882.

[53] D. Faustino, N. Gonçalves, M. Portela, and A. Rito Silva, "Stepwise migration of a monolith to a microservice architecture: Performance and migration effort evaluation," Performance Evaluation, vol. 164, p. 102411, May 2024, doi: 10.1016/j.peva.2024.102411.

[54] K. Borowa, A. Ratkowski, and R. Verdecchia, "The Technical Debt Gamble: A Case Study on Technical Debt in a Large-Scale Industrial Microservice Architecture," 2024, doi: 10.2139/ssrn.4946579.

[55] M. Ait Said, L. Belouaddane, S. Mihi, and A. Ezzati, "A Systematic Framework To Enhance Reusability In Microservice Architecture," IJCDS, vol. 17, no. 1, pp. 1–18, Jan. 2025, doi: 10.12785/ijcds/1571009176.

[56] J. Daniel, G. Mota, X. Wang, and E. Guerra, "Architecture Refactoring Towards Service Reusability in the Context of Microservices," in Agile Processes in Software Engineering and Extreme Programming, S. Peter, M. Kropp, A. Aguiar, C. Anslow, M. I. Lunesu, and A. Pinna, Eds., in Lecture Notes in Business Information Processing, vol. 545. Cham: Springer Nature Switzerland, 2025, pp. 129–144. doi: 10.1007/978-3-031-94544-1_9.

[57] G.-D. Schwarz, P. Heltweg, and D. Riehle, "Balancing technology heterogeneity in microservice architectures," Empir Software Eng, vol. 30, no. 5, p. 127, Sep. 2025, doi: 10.1007/s10664-025-10684-4.

[58] G. Schwarz, F. Quast, and D. Riehle, "Ensuring Syntactic Interoperability Using Consumer-Driven Contract Testing," Software Testing Verif &amp; Rel, vol. 35, no. 5, p. e70006, Aug. 2025, doi: 10.1002/stvr.70006.

[59] M. Zhang et al., "Fuzzing microservices: A series of user studies in industry on industrial systems with EvoMaster," Science of Computer Programming, vol. 246, p. 103322, Dec. 2025, doi: 10.1016/j.scico.2025.103322.

[60] D. Badampudi, M. Usman, and X. Chen, "Large scale reuse of microservices using CI/CD and InnerSource practices - a case study," Empir Software Eng, vol. 30, no. 2, p. 41, Mar. 2025, doi: 10.1007/s10664-024-10595-w.

[61] U. Veeramreddygari, "Retail Digital Transformation Through Microservices and Containerized Architectures," in 2025 6th International Conference on Data Intelligence and Cognitive Informatics (ICDICI), Tirunelveli, India: IEEE, Jul. 2025, pp. 1531–1536. doi: 10.1109/ICDICI66477.2025.11135409.

[62] E. Evans, Domain-driven design: tackling complexity in the heart of software. 2004.