IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Ingénierie des Systèmes d'Information
Vol. 31, No. 5, May, 2026, pp. 1685-1694
Journal homepage: http://iieta.org/journals/isi

Empirical Insights into the Synergy of Extreme Programming, DevOps, and Microservices

Nagalambika Swamy[1,2*], L. Manjunatha Rao[1]

[1] Department of Master of Computer Applications, Dr. Ambedkar Institute of Technology, Bangalore 560056, India
[2] Department of Master of Computer Applications, St. Claret College, Bangalore 560013, India

Corresponding Author Email: nagalambika.swamy@gmail.com

https://doi.org/10.18280/isi.310524

Received: 5 March 2026
Revised: 25 April 2026
Accepted: 8 May 2026
Available online: 31 May 2026

Keywords:
agile, DevOps, Extreme Programming, Microservice Architecture, distributed software development, continuous integration and delivery, software process integration, practitioner survey

ABSTRACT

This study develops and evaluates a conceptual framework that integrates Extreme Programming (XP), DevOps, and Microservice Architecture (MSA). The framework aligns XP practices with service-oriented decomposition, continuous integration and continuous delivery pipelines, independent release management, and continuous monitoring. Its feasibility was assessed through a practitioner survey using an online order-management case study comprising four independent services. Seventy IT professionals with experience in agile development, DevOps, or microservices evaluated system suitability, distributed-team practices, development planning, release management, documentation, and component reusability. Likert-scale items were analysed using one-sample t-tests against the neutral midpoint, whereas categorical planning and release-management items were examined using chi-square tests. The instrument demonstrated good internal consistency (Cronbach's $\alpha = 0.85$). Participants reported strong perceived suitability for large-scale, complex, and geographically distributed software projects, with mean ratings between 4.00 and 4.21 for scalability, reusability, deployment readiness, development efficiency, and software quality (all $p < 0.001$). XP practices for geographically distributed teams received mean ratings of 4.25–4.45, while planning and release assessments showed statistically significant role-related response patterns. The findings provide preliminary evidence that the proposed framework is perceived as a feasible process model for large-scale, complex, and distributed software development. Implementation-based studies using deployment frequency, Lead Time for Changes (LTC), coupling, reuse, and reliability metrics are needed to establish operational effects.

1. INTRODUCTION

Modern software systems are becoming increasingly large, distributed, and continuously evolving. Agile methods, particularly Extreme Programming (XP), have brought tremendous improvement in communication, coding disciplines, and iterative delivery [1]. However, XP was originally designed for small, co-located teams, and remains largely code-centric, providing limited guidance on architectural planning, distributed collaboration, and large-scale deployment.

Due to these limitations of XP, recent research has explored extending and adapting XP for complex systems, microservice-based applications, and distributed development environments. However, these efforts remain fragmented [2]. The goal of this integration is to better support medium-to-large-scale, distributed, and design-intensive software projects compared with existing fragmented approaches.

To address architectural challenges that XP alone cannot adequately support, Microservice Architecture (MSA) has emerged as a leading architectural paradigm for building modular, scalable, and independently deployable services [3-5]. Industry leaders such as Netflix, Amazon, and Uber have widely adopted microservices due to their flexibility and resilience in large-scale systems [6].

The literature emphasizes the usefulness of microservices in the system of telecommunications, NFV environments, SaaS product lines, and IoT systems [7, 8]. Microservice approaches have also been extended to fog-cloud and IoT domains that support distributed and low-latency workloads [9] introducing a microservice-based industrial IoT system within a fog-cloud assisted network, and showing practical deployment patterns. Nevertheless, these studies have focused on architectural benefits, and provide limited discussion on the integration of XP practices within microservice-driven development.

DevOps practices complement microservices through automated software delivery pipelines and improve the collaboration between development and operations teams [10]. DevOps is an essential part of the efficient deployment of large-size systems by emphasizing the continuous integration and continuous delivery (CI/CD), containerization and operational monitoring. Numerous empirical studies have shown that automated pipelines, test-driven methods, and container orchestration platforms such as Docker and Kubernetes have enhanced the reliability, scalability, and coordination of the deployment pipeline for teams [11, 12]. Moreover, DevOps also increases the productivity of geographically dispersed teams by minimizing the

communication overhead caused by automation [13]. However, most DevOps research focuses on operational automation and infrastructure management, with limited attention paid to XP's engineering practices and development discipline.

Despite substantial research on XP, DevOps, and MSA individually, existing studies primarily investigate these approaches independently or focus on limited pairwise integrations such as XP with BDD or microservices within cloud-native environments [14-18]. Consequently, the combined integration of XP, DevOps, and MSA remains insufficiently explored, particularly in the context of large-scale and geographically distributed software development projects.

1.1 Research gap and study motivation

Existing research primarily investigates XP, DevOps, and MSA independently or through limited pairwise integrations. Studies on XP focus on agile development practices, DevOps emphasizes deployment automation and operational efficiency, while MSA research concentrates on scalability, modularity, and service independence [1-18].

Recent work has explored combinations such as XP with Behavior-Driven Development (BDD) and microservices within cloud-native environments [14-18]. However, these studies address specific aspects of software development and do not provide a unified framework that simultaneously integrates agile engineering practices, deployment automation, and microservice-based architectural design.

Furthermore, there is limited empirical evidence regarding the feasibility of integrating XP, DevOps, and MSA into a single framework for large-scale and geographically distributed software projects. Consequently, there is limited understanding of whether such an integrated approach can effectively support scalability, distributed-team collaboration, development planning, documentation, and component reusability.

To address this gap, this study proposes an integrated XP–DevOps–MSA framework and evaluates its feasibility through a practitioner-based empirical survey. In addition, a technical evaluation framework is introduced to support future implementation-based validation.

1.2 Theoretical foundation of XP–DevOps–MSA integration

XP, DevOps, and MSA are grounded in distinct yet complementary software engineering principles. XP emphasizes lightweight development, minimal documentation, continuous feedback, and rapid iteration. In contrast, MSA promotes modular decomposition, independent deployment, and well-defined service boundaries, often requiring detailed interface specifications and infrastructure planning. DevOps bridges development and operations through automation, CI/CD, and monitoring.

Despite their individual strengths, integrating these paradigms introduces inherent methodological tensions. XP advocates minimal upfront design, whereas microservices demand careful architectural planning, including service granularity, API contracts, and inter-service communication mechanisms. Similarly, DevOps requires structured pipelines and operational governance, which may conflict with XP's informal and flexible practices.

This research is based on the premise that these methodological differences can be systematically reconciled through a unified framework that aligns XP's iterative development practices with DevOps automation and microservices-based modular architecture. By harmonizing these paradigms, the proposed model aims to achieve a balance between agility, scalability, and operational efficiency.

1.3 Problem statement

Existing research presents only partial integration: XP paired with CI but not Microservices; DevOps combined with microservices but without XP planning practices; or Agile–DevOps workflows lacking architectural decomposition. Microservice studies focus on scalability and deployment, but do not connect these benefits to XP workflows.

The problem addressed in this study is the absence of a conceptual software development process that integrates XP, DevOps, and MSA for medium- and large-scale complex software projects. No prior work has provided a comprehensive end-to-end model combining XP, DevOps, and MSA, nor an empirical validation of its practicality, as most studies have examined these approaches in isolation or in partial pairings.

1.4 Contributions of the study

This study contributes to software engineering research in three ways. First, it proposes a unified XP–DevOps–MSA framework that defines how agile engineering practices, DevOps automation, and microservice-based architectural principles interact throughout the software development lifecycle. Unlike prior studies that examine these approaches independently or in partial combinations, the framework provides an integrated process model supported by defined lifecycle phases and design principles.

Second, the study addresses the gap between XP's lightweight development philosophy and the architectural and operational demands of microservice-based systems by combining iterative development, automated delivery pipelines, and modular service-oriented design within a single process.

Third, the framework is evaluated through a practitioner-based feasibility assessment using a microservices-oriented case study, providing statistical evidence of its applicability to scalability, distributed development, architectural planning, documentation, deployment readiness, and component reusability.

Collectively, these contributions advance the understanding of XP–DevOps–MSA integration and provide a foundation for future implementation-based and industrial evaluation studies.

1.5 Study objectives and research questions

The objectives of this study are as follows:

(1) To develop a conceptual software development process that integrates XP, DevOps, and MSA for medium- and large-scale complex software projects.
(2) To evaluate whether the proposed process supports geographically distributed software development teams.
(3) To evaluate whether the proposed process supports both code-centric and design-centric software

development activities.

(4) To evaluate whether the proposed process improves architectural design activities and associated software documentation.

(5) To evaluate whether the proposed process promotes software component reusability.

To address these objectives, we formulated the following research questions

- RQ1: How feasible is the proposed XP–DevOps–MSA framework for executing medium- to large-scale complex and geographically distributed software projects?
- RQ2: How effectively does the framework support architectural planning, release management, and documentation?
- RQ3: To what extent does the framework support modularity and component reusability?

To address these research questions, three specific hypotheses (H1-H3) were formulated to evaluate individual dimensions of the proposed framework:

(1) H1: The integrated XP–DevOps–MSA framework significantly improves effectiveness in handling medium- to large-scale complex projects.

(2) H2: The framework effectively supports geographically distributed software development.

(3) H3: The framework facilitates comprehensive architectural planning, documentation, and component reusability.

The hypotheses operationalize specific dimensions of the broader research questions and enable statistical evaluation of practitioner perceptions regarding the proposed framework.

2. CONCEPTUAL XP–DEVOPS–MSA FRAMEWORK

This section presents the proposed XP–DevOps–MSA integrated framework as a unified software development process. It describes the conceptual model, underlying design principles, and process lifecycle that combine XP practices, DevOps automation, and microservice-based architecture. The framework aims to support scalable, distributed, and high-quality software development.

2.1 The conceptual software development process

The proposed framework integrates XP engineering practices, MSA, and DevOps-enabled CI/CD pipelines into a unified software development process. Figure 1 illustrates how these three paradigms interact throughout the software development lifecycle and collectively support modularity, scalability, and continuous delivery.

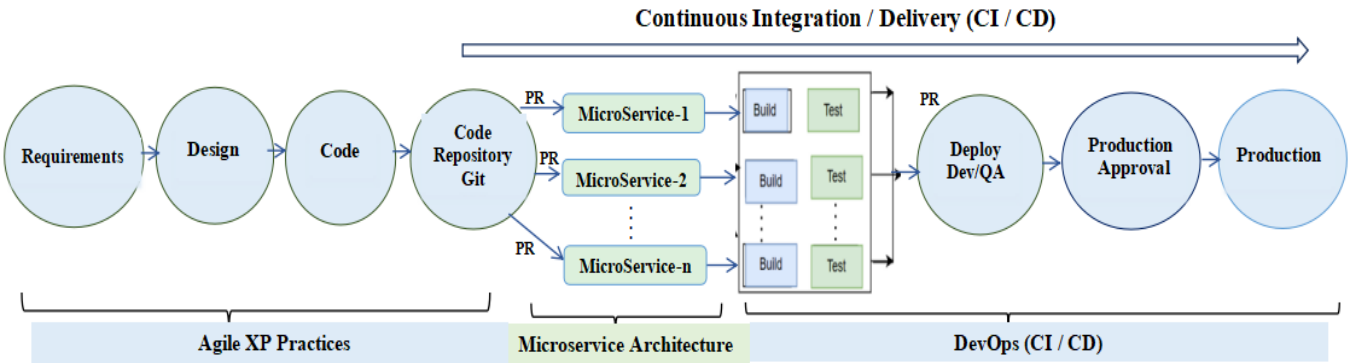

Figure 1. Conceptual XP–DevOps–MSA framework for software development

Table 1. Integrated XP–DevOps–MSA software development process lifecycle

Lifecycle Phase	XP Practices	DevOps Activities	MSA Considerations	Key Artifacts Produced
Requirements Planning	User stories, customer collaboration, planning game	Pipeline readiness planning	Identification of candidate microservices	Product backlog, service backlog
Architectural Design	Simple design, collective ownership	Infrastructure planning	Service boundary definition, API contracts	Service design documents, API specifications
Development	Pair programming, coding standards	Automated builds triggered by commits	Independent service implementation	Source code repositories
Testing	Test-driven development (TDD), acceptance testing	Automated unit and integration testing	Service-level test isolation	Test reports, coverage metrics
Integration	Continuous integration	CI pipelines, container builds	Independent service integration	Build artifacts, container images
Release Planning	Iteration planning	Continuous delivery pipelines	Versioned service releases	Release plans, deployment scripts
Deployment	Frequent small releases	Automated deployment to environments	Independent service deployment	Deployment logs
Monitoring & Feedback	Continuous feedback	Monitoring, logging, rollback support	Service health monitoring	Performance metrics, feedback reports

Figure 1 presents the integration of XP, MSA, and DevOps within a unified software development process. XP supports requirements analysis, design, coding, and iterative feedback, while MSA organizes functionality into independently deployable services. DevOps provides automated CI/CD pipelines for building, testing, deploying, and monitoring these services. Together, the three approaches combine agile development, architectural modularity, and deployment automation to support scalable and geographically distributed software projects.

The integrated lifecycle phases of the proposed framework are summarized in Table 1, which illustrates how XP practices, DevOps automation activities, and MSA considerations are aligned across the software development lifecycle. The framework also serves as the conceptual basis for the survey instrument and feasibility assessment conducted in this study.

Design principles

To guide the practical implementation of the integrated framework, four fundamental design principles underpin the proposed process.

P1: Service-aligned XP Iterations

XP iterations are organized around individual microservices, ensuring that practices such as pair programming and test-driven development remain focused within service boundaries and support modular distributed development.

P2: Pipeline-Driven Feedback Loops

Automated CI/CD pipelines integrate build, test, deployment, and monitoring activities to provide continuous feedback and accelerate validation through DevOps automation.

P3: Architecture-First Modular Decomposition

The system is decomposed into microservices aligned with business capabilities, providing a foundation for iteration planning and scalable, maintainable development.

P4: Independent Release Governance

Each microservice follows an independent release and versioning strategy, enabling autonomous deployment, reducing coordination effort, and supporting rapid delivery.

3. RESEARCH METHODOLOGY

This study adopts a quantitative descriptive research design, supplemented by qualitative insights obtained from open-ended practitioner responses. A case study based on an Online Order Management System was used because of confidentiality and intellectual property restrictions associated with industrial datasets. A structured questionnaire was used to assess the suitability of practices for complex software development. The evaluation focuses on practitioner perceptions of the proposed framework, informed by findings reported in existing literature on standalone XP, DevOps, and Microservice-based approaches.

The objective of this study is not to experimentally benchmark the integrated framework against existing development approaches, but rather to evaluate its perceived feasibility and applicability through practitioner-informed assessment. The proposed XP–DevOps–MSA framework is therefore investigated as a conceptual process model within a controlled case-study context. Future work will focus on implementing the framework in industrial projects and measuring objective performance indicators such as deployment frequency, defect density, lead time, and service reliability.

3.1 Research design

This study employs a descriptive survey-based empirical research approach to evaluate the feasibility of the proposed XP–DevOps–MSA framework from a practitioner perspective. The approach is appropriate because the objective is to assess practitioner perceptions and feasibility rather than establish causal relationships through controlled experimentation.

Although primarily quantitative, the study incorporates qualitative insights through open-ended responses to capture practitioner observations, challenges, and expectations. This mixed approach complements the statistical findings with contextual explanations.

To provide a realistic evaluation context, participants were presented with an Online Order Management System case study comprising multiple microservices and independent data management components. The case study served as the reference model for assessing system suitability, XP practices, sprint feasibility, and release planning activities.

3.2 Case study: Online Order Management System

The questionnaire was developed based on the proposed XP–DevOps–MSA conceptual framework and the Online Order Management System case study to evaluate the feasibility and applicability of the integrated approach.

A distributed Online Order Management System was used as the contextual basis of this research. The system is composed of four independent microservices: Account, Product, Cart, and Order, each with its own database and supported by Redis as a caching layer to handle incoming requests. The architecture reflects a realistic microservices environment comprising modular components, independent deployment, and CI/CD-enabled updates.

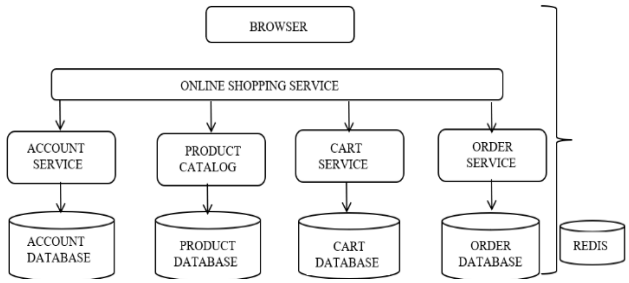

Figure 2. Online Order Management System architecture

Figure 2 shows the Online Order Management System consisting of four independent services—Account, Product, Cart, and Order microservices with independent databases and Redis as a caching layer that improves system performance by reducing database access latency and efficiently handling high volumes of incoming requests. The architecture reflects a realistic microservices environment suitable for evaluating scalability, modularity, deployment independence, and component reuse.

3.2.1 Case study utilization

The Online Order Management System was used as the reference application scenario for evaluating the proposed XP–DevOps–MSA framework. The case study incorporates characteristics of medium- to large-scale distributed applications, including multiple business services, independent deployment requirements, scalability demands, and cross-functional team collaboration. It provided a common project context for all survey participants, enabling consistent evaluation of framework characteristics such as scalability, modularity, deployment readiness, architectural flexibility, and component reusability.

The framework phases, development activities, and

questionnaire items were derived and interpreted within this case-study environment, thereby reducing interpretation variability among respondents. Consequently, the case study functioned as a structured evaluation and contextual validation mechanism supporting the practitioner-based feasibility assessment rather than serving solely as background information.

3.3 Population and sampling

This study surveyed 70 IT professionals with experience in Agile development (particularly XP), DevOps, and MSA. The demographic profile of the participants is presented in Table 2. Participants were required to have prior exposure to at least one of these approaches and a minimum of five years of industry experience to ensure response validity. Stratified sampling was used to obtain perspectives from managers, developers, team leaders, and architects involved in software design, development, testing, and deployment. Participants were recruited through professional networks, including LinkedIn, ResearchGate, Google Scholar profiles, Facebook groups, and industry contacts.

The sample size is considered adequate for exploratory empirical software engineering research and supports the statistical analyses conducted in this study.

Table 2. Sampling frame (N = 70)

Sl. No	Role	Experience (in Years)	Frequency	Percentage (%)
1	Manager	15-22	27	38.57%
2	Developer	5-17	16	22.86%
3	Team Leader	6-15	18	25.71%
4	Architect	6-17	09	12.86%

3.4 Questionnaire design

A structured questionnaire was developed to evaluate practitioner perceptions of the integrated XP–DevOps–MSA framework using the Online Order Management case study. The questionnaire was designed based on case study observations and expert input from IT professionals experienced in distributed microservice systems. Sections Q8 and Q9 were derived from the 21-day sprint and release scenario to enable realistic assessment of development and delivery expectations. The questionnaire structure is summarized in Table 3.

The instrument consisted of five sections:

(1) Section-1 (Q1–Q6): Six 5-point Likert-scale items evaluating scalability, component reusability, loose coupling, global deployment readiness, development speed, and software quality.
(2) Section-2 (Q7): Assessment of key XP practices, including daily stand-ups, adaptive planning, code control, continuous integration, visual indicators, and Code Gallery in distributed microservice environments.
(3) Section-3 (Q8): Nine Yes/No items evaluating sprint-level indicators such as iterations, architectural changes, collaboration, velocity, and related project activities.
(4) Section-4 (Q9): Four Yes/No items assessing release-related outcomes, including code exhibits, completed user stories, user story growth, and story points.
(5) Section 5: Open-ended questions capturing challenges, risks, and improvement suggestions.

The questionnaire was designed to align directly with the study objectives, research questions, and hypotheses.

Table 3. Practitioner questionnaire on XP–DevOps–MSA integration

Q. No	Section / Question Description	Response Type
	System Suitability Assessment — (Q1-Q6)	
Q1	Is Microservice Architecture suitable for large and distributed applications?	
Q2	Does the above process support component reusability?	
Q3	Does the above process support loosely coupled design?	
Q4	Does the above process support deployment across the globe?	
Q5	Does the above process improve development speed and cost efficiency?	5-point Likert
Q6	Does the above process improve the quality of applications?	scale
	XP Practice Evaluation — (Q7)	
Q7	XP practices implemented in the application: Daily stand-up meetings, Adaptive planning, Continuous Integration, Code Control, Code Gallery, XP Project management, Visual Indicators - Are these practices necessary for distributed applications?	
	Full Development Plan — (Q8)	
Q8.1	Number of hours spent in training and upgrades- 40 hours - 8 hours each day for 5 days	
Q8.2	Number of iterations for 4 releases	
Q8.3	Does the application support architectural design changes?	
Q8.4	Team collaboration level (High/Low)	
Q8.5	Satisfaction with working environment/culture	Yes/No
Q8.6	Interest or buy-in level	
Q8.7	Velocity increase (speed of deliverables)	
Q8.8	Use of Code Gallery–5 working days, 8 hrs./day	
Q8.9	Weekly work hours – 40 hours, 8 hrs./day	
	Release Plan Evaluation — (Q9)	
Q9.1	Code exhibit counts per release	
Q9.2	Number of user stories completed per release	Yes/No
Q9.3	Percentage increase in user stories per release	
Q9.4	Story points per release	
	Suggestions	Text
	Open– ended feedback on XP–DevOps–icroservices integration	

3.5 Measurement considerations

The study evaluates the perceived feasibility and applicability of the proposed XP–DevOps–MSA framework through practitioner assessments rather than a full-scale industrial implementation. Accordingly, constructs such as scalability, deployment readiness, architectural modularity, software quality, and component reusability were assessed using five-point Likert-scale responses from experienced software practitioners.

To strengthen the assessment, the questionnaire included project-oriented indicators such as release planning, iteration management, architectural change support, velocity improvement, and documentation practices, reflecting common activities in agile, DevOps, and microservice-based projects.

In addition, objective software engineering metrics—Coupling, Component Reuse Ratio (CRR), Deployment Frequency (DF), and Lead Time for Changes (LTC)—were defined to support future implementation-based validation. These metrics were not directly measured in the present study but provide a basis for quantitative evaluation in real-world deployments.

Therefore, the findings should be interpreted as evidence of practitioner-perceived feasibility rather than direct measurements of technical performance.

3.6 Data analysis

Descriptive statistics (means and standard deviations) were computed for all questionnaire items across practitioner roles. For Likert-scale items (Q1–Q7), one-sample t-tests were applied to the combined responses (Architects, Developers, Managers, Team Leads) to examine whether mean ratings significantly differed from the neutral midpoint value of 3 on a 5-point scale. For categorical (Yes/No) items (Q8 and Q9), chi-square tests of independence were used to examine associations between participant designation and agreement patterns. The null hypothesis (H_0) assumed no significant difference or association, while the alternative hypothesis (H_1) indicated significant differences. A significance threshold of $\alpha = 0.05$ was maintained for all statistical tests.

3.7 Reliability and validity

Cronbach's alpha was used to assess the reliability of the survey instrument. The seven Likert-based items (Q1-Q7) measuring the perceived suitability of XP–DevOps–MSA integration were found to be highly internally consistent ($\alpha = 0.85$). As Q8 and Q9 comprised a binary approach i.e. yes/no responses, reliability could not be assessed using Cronbach's alpha technique, as shown in Table 4.

Table 4. Reliability analysis of questionnaire blocks using Cronbach's alpha

Questionnaire Block	Number of Items	Measurement Type	Cronbach's α	Reliability Level
Q1 – Q7 (System Suitability Factors & XP Practices)	7	Likert (1–5)	0.85	High Reliability
Q8, Q9 (Project Performance & Release Plan)	9	Yes/No	Not Applicable	—

3.8 Ethical considerations

Participant confidentiality was preserved throughout the research study. All responses were anonymized and the data were processed according to ethical research guidelines.

4. RESULTS AND DISCUSSION

This section presents the analysis and findings for the three research hypotheses by using descriptive statistics, one-sample t-test and chi-square test. Statistical significance was tested at $\alpha = 0.05$ using one-sample t-tests and chi-square tests. The one-sample t-tests compared practitioner ratings against the neutral midpoint value of 3 on the five-point Likert scale. Given the sample size of 70 respondents, parametric testing was considered appropriate for evaluating overall practitioner agreement. Effect sizes (Cohen's d) were also reported to assess practical significance.

The questionnaire responses were collected using five-point Likert-scale items. Although Likert-scale data are ordinal in nature, one-sample t-tests were employed because the study aimed to assess whether the mean response differed significantly from the neutral midpoint value (3). The use of parametric tests for Likert-scale data is widely accepted in empirical software engineering research when the scale contains five or more response categories and the sample size is sufficiently large. With 70 respondents, the sample size satisfies the conditions under which the sampling distribution of the mean can be approximated as normal according to the Central Limit Theorem.

Accordingly, the statistical results should be interpreted as evidence of practitioner consensus regarding the feasibility and perceived benefits of the proposed framework, rather than as direct measurements of operational or system-level performance.

4.1 Hypothesis 1: The integrated XP–DevOps–MSA framework significantly improves effectiveness in handling medium- to large-scale complex projects

Hypothesis 1 was assessed using one-sample t-tests on system suitability items (Q1-Q6). Table 5 reveals that integrating XP, DevOps, and MSA significantly enhances scalability, modularity, reusability, deployment readiness, development efficiency, and software quality, with mean ratings from 4.00-4.21 ($p < 0.001$, large effect sizes).

These findings indicate strong practitioner agreement regarding the suitability of the framework for medium- to large-scale complex software projects.

The strong support for Hypothesis 1 can be attributed to the complementary strengths of the three integrated approaches. MSA enables modular decomposition of complex systems into independently deployable services, improving scalability and maintainability. XP contributes iterative development, continuous feedback, and engineering practices that enhance development quality and responsiveness to changing requirements. DevOps introduces automation through continuous integration and continuous delivery pipelines, reducing deployment effort and improving release reliability. The combination of these capabilities enables the framework to address both development and operational challenges

associated with medium- to large-scale distributed software projects.

Hence, Hypothesis 1 is strongly supported.

Table 5. System suitability factors (Q1–Q6) and XP Practices (Q7) using one-sample t-tests

Questionnaire Item	Mean± SD	T-Value (df = 69)	P-Value	Cohen's d
Q1: Suitability for large, distributed apps	4.21 ± 0.48	22.35	<0.001	2.67
Q2: Component reusability	4.15 ± 0.50	20.12	<0.001	2.41
Q3: Loosely coupled design	4.10 ± 0.52	18.52	<0.001	2.22
Q4: Deployed across globe	4.05 ± 0.55	17	<0.001	2.03
Q5: Development speed and cost efficiency	4.00 ± 0.57	15.78	<0.001	1.89
Q6: Improvement in quality of applications	4.12 ± 0.49	19	<0.001	2.28

4.2 Hypothesis 2: The framework effectively supports geographically distributed software development

Hypothesis 2 was tested using the seven XP-practice sub-items of Q7, as shown in Table 6. One-sample t-tests on the XP practice items (Q7 sub-items) showed consistently high mean ratings (4.25–4.45), all significantly above the neutral midpoint of 3 ($p < 0.001$), with very large effect sizes (Cohen's $d > 1.8$). This indicates a strong practical and statistical significance across all seven XP practices, demonstrating that practitioners view these practices as essential for coordinating work in geographically distributed microservices environments.

Table 6. Evaluation of XP practices for geographically distributed projects (Q7)

Q7. XP Practice	Mean ± SD	T-Value (df = 69)	P-Value	Cohen's d
Daily Stand-up meetings	4.41 ± 0.52	18.45	<0.001	2.2
Adaptive planning	4.38 ± 0.50	17.8	<0.001	2.12
Code Control	4.32 ± 0.53	16.25	<0.001	1.94
Continuous Integration	4.45 ± 0.48	19.1	<0.001	2.27
XP Project Management	4.28 ± 0.52	15.85	<0.001	1.89
Visual Indicators	4.25 ± 0.54	15.3	<0.001	1.82
Code Gallery	4.40 ± 0.50	18	<0.001	2.14

The positive evaluation of XP practices in geographically distributed environments is primarily driven by their emphasis on communication, visibility, and continuous coordination. Daily stand-up meetings, adaptive planning, visual indicators, and continuous integration provide mechanisms for maintaining team alignment despite geographical separation. Practitioners particularly valued continuous integration and code-sharing practices because they reduce integration conflicts and improve collaboration among distributed development teams.

Therefore, Hypothesis 2 is confirmed.

4.3 Hypothesis 3: The framework facilitates comprehensive architectural planning, documentation, and component reusability

Hypothesis 3 was evaluated by combining evidence from the Full Development Plan (Q8) and Release Plan (Q9) assessments. Sub-items in Q8 measured design-centric and documentation-related aspects such as architectural design change support (Q8.3), iteration planning (Q8.2), measurable velocity (Q8.7), and Code Gallery usage (Q8.8), which serves as a documentation repository. Chi-square tests of independence (Tables 7 and 8) revealed significant associations for these design-centric aspects, as well as for code-centric delivery metrics including completed user stories and story points.

Table 7. Chi-square results for Full Development(q8) and Release Plan (Q9)

Questionnaire Item(s)	χ^2 Statistic	df	P-Value
Q8. Full Development Plan	12.78	3	0.005
Q9. Release Plan	9.54	3	0.023

Table 8. Chi-square result for Full Development Plan (Q8)

Questionnaire Item(s)	χ^2 Statistic	df	P-Value
Q8-Full Development Plan			
Q8.1 Numbers of Hours Spent in Training and Upgrades	4.32	3	0.229
Q8.2 Number of Iterations for 4 releases	7.85	3	0.049
Q8.3 Application Architectural Design Change support	10.12	3	0.018
Q8.4 Team Collaboration	5.47	3	0.14
Q8.5 Satisfaction (working environment/Culture)	3.29	3	0.35
Q8.6 Interest (buy-in)	6.15	3	0.104
Q8.7 Velocity Increase (Speed of Deliverable)	8.56	3	0.036
Q8.8 Code Gallery	9.02	3	0.029
Q8.9 Weekly Work Hours	2.89	3	0.409

As shown in Table 7, both the Full Development Plan (Q8) and Release Plan (Q9) demonstrated statistically significant results ($p < 0.05$), indicating practitioner agreement that the proposed framework supports structured planning and release management activities. The detailed Q8 analysis in Table 8 further revealed significant support for architectural design changes (Q8.3, $p = 0.018$), iteration planning (Q8.2, $p = 0.049$), velocity improvement (Q8.7, $p = 0.036$), and Code Gallery usage (Q8.8, $p = 0.029$), suggesting that the framework effectively supports architectural planning, documentation, and development coordination. In addition, component reusability (Q2) achieved strong practitioner agreement (Mean = 4.15, $p < 0.001$) as reported in Table 5.

These findings can be attributed to the complementary strengths of the integrated approaches: MSA promotes upfront architectural planning through service decomposition and API design, XP supports iterative development and continuous refinement, while DevOps enables automated testing, integration, and deployment. Together, these capabilities

balance design-centric and code-centric activities, facilitate documentation and release management, and enhance component reuse throughout the software development lifecycle. Therefore, Hypothesis 3 is supported.

4.4 Respondent perspectives and observations

Qualitative feedback was collected through the open-ended suggestion section of the questionnaire to capture practitioner perspectives on the proposed integrated software development process.

Feedback indicated that integrating MSA with DevOps improves application quality, scalability, and modularity through better testing and deployment practices. Respondents noted that XP supports distributed development through iterative delivery and loosely coupled services.

Challenges included training requirements, workload management, and increased pressure from sprint deadlines, which may affect development and testing activities. Effective sprint planning, clear user stories, and consideration of team availability were identified as important factors for maintaining project progress. Overall, respondents viewed the integrated approach positively while acknowledging associated workload and coordination challenges.

4.5 Technical evaluation framework and metric applicability

To complement the practitioner-based survey evaluation, a technical evaluation framework was developed to identify objective software engineering metrics that can be used to assess the effectiveness of the proposed XP–DevOps–MSA framework. These metrics were selected because they directly correspond to the framework objectives of scalability, modularity, deployment readiness, software quality, and component reusability.

Although the present study does not include a full implementation-based experiment and therefore does not directly measure these metrics, their inclusion establishes a traceable linkage between practitioner-evaluated framework characteristics and quantifiable software engineering outcomes. Consequently, the proposed metrics framework serves as a bridge between the current feasibility assessment and future implementation-based validation studies, providing a structured basis for objective comparison between standalone XP, DevOps, MSA, and the proposed integrated framework.

4.5.1 Objective technical metrics

Recent studies have highlighted the importance of reusable architectural patterns, automation, continuous delivery, and collaborative practices in supporting real-time IoT environments, distributed service systems, and modern software delivery processes [19, 20].

Four objective software engineering metrics were selected from established literature software architecture, software reuse, and DevOps performance measurement [21-23]. Coupling evaluates architectural modularity and service independence in microservice-based systems, while CRR measures reusable component utilization. DF and LTC, adopted from the DevOps Research and Assessment (DORA) framework, assess software delivery capability and deployment efficiency. Together, these metrics align with the primary objectives of the proposed XP–DevOps–MSA framework: scalability, modularity, reusability, deployment readiness, and development efficiency. The following formulations illustrate their potential application in future implementation-based evaluations.

(1) Coupling Metric (C): Measures the degree of dependency among microservices.

$$C = \frac{\text{Number of Inter} - \text{Service Calls}}{\text{Total Possible Service Interactions}} \quad (1)$$

Lower values indicate greater service independence and modularity.

(2) Component Reuse Ratio (CRR): Measures the proportion of reusable software components.

$$CRR = \frac{\text{Reused Components}}{\text{Total Components}} \quad (2)$$

Higher values indicate improved software reuse.

(3) Deployment Frequency (DF): Measures the number of successful deployments within a specified period.

$$DF = \frac{\text{Number of Deployments}}{\text{Time Period}} \quad (3)$$

Higher values indicate greater delivery agility and deployment readiness.

(4) LTC: Measures the elapsed time between code commit and deployment.

$$\text{LTC} = \text{Deployment Time} - \text{Commit Time} \quad (4)$$

Lower values indicate faster software delivery and feedback cycles.

4.5.2 Mapping framework objectives to metrics

The selected metrics were mapped to the framework objectives based on their ability to quantify key characteristics of distributed software systems. Coupling (C) was associated with scalability and architectural modularity because loosely coupled services facilitate independent scaling and reduce inter-service dependencies. CRR was selected to assess the framework's support for reusable software assets. DF and LTC were chosen to evaluate deployment readiness and development agility, as they are widely recognized indicators of software delivery performance. The relationship between framework objectives and the corresponding technical metrics is summarized in Table 9. Together, these metrics provide a quantitative foundation for future implementation-based validation of the proposed XP–DevOps–MSA framework.

Table 9. Mapping framework objectives to technical metrics

Framework Objective	Technical Metric
Scalability	Coupling (C), DF
Component Reusability	CRR
Distributed Development	DF
Deployment Readiness	DF, LTC
Software Quality	LTC
Architectural Modularity	Coupling

4.5.3 Case-study-based metric applicability

The Online Order Management System, consisting of independent Account, Product, Cart, and Order microservices, provides a representative environment for applying the

proposed metrics. Coupling can be assessed through inter-service communication, CRR through shared reusable modules, and DF and LTC through CI/CD pipeline activities. The case study therefore demonstrates the applicability of the proposed metrics for future implementation-based validation.

4.5.4 External validity considerations

The findings of this study should be interpreted within the context of its external validity constraints. The practitioner sample consisted of 70 software professionals who evaluated the proposed XP–DevOps–MSA framework using a common case-study scenario. Although the participants represented different software development roles and levels of experience, the sample size and geographic coverage may not fully represent the diversity of software engineering practices across industries, organizations, and regions.

Furthermore, the evaluation was based on practitioner perceptions regarding the feasibility and applicability of the proposed framework rather than direct implementation-based measurements. Consequently, the results provide evidence of practitioner consensus and perceived usefulness but should not be generalized as definitive performance outcomes for all software development environments.

Nevertheless, the inclusion of experienced practitioners, the use of a common microservices-oriented case study, and the statistical significance observed across multiple evaluation dimensions provide reasonable preliminary evidence supporting the framework's applicability. Future studies involving larger and more geographically diverse samples, together with implementation-based validation, are required to strengthen the generalizability of the findings.

4.6 Discussion

The comparison presented in Table 10 is derived from the characteristics of XP, DevOps, and MSA reported in the literature reviewed in Section 1. The table provides a conceptual synthesis of the strengths and limitations of the individual approaches and illustrates how the proposed XP–DevOps–MSA framework combines their complementary capabilities. The comparison is intended as a conceptual analysis based on existing literature rather than an empirical performance evaluation.

Table 10. Conceptual comparison of existing approaches and proposed framework

Capability	XP	DevOps	MSA	Proposed XP–DevOps–MSA
Iterative Development	Strong	Limited	Not Primary Focus	Strong
Customer Feedback	Strong	Limited	Not Primary Focus	Strong
Continuous Integration/Delivery	Limited	Strong	Limited	Strong
Deployment Automation	Not Primary Focus	Strong	Limited	Strong
Architectural Modularity	Limited	Not Primary Focus	Strong	Strong
Independent Service Deployment	Not Primary Focus	Limited	Strong	Strong
Support for Distributed Teams	Limited	Moderate	Moderate	Strong
Reusability Support	Limited	Not Primary Focus	Moderate	Strong
Large-scale Project Support	Limited	Moderate	Strong	Strong

Table 10 highlights the complementary strengths of XP, DevOps, and MSA. XP supports iterative development and customer collaboration, DevOps provides CI/CD automation, and MSA enables modularity, scalability, and independent deployment. By integrating these capabilities, the proposed XP–DevOps–MSA framework offers a unified approach for scalable development, deployment automation, architectural modularity, and distributed-team collaboration.

4.7 Limitations and future research

The study involved 70 software professionals, which provides useful practitioner insights but may not fully represent the diversity of software engineering practices across different industries, organizational contexts, and geographic regions. Future studies should involve larger scale of studies at various industries as well as geographical areas.

Recent studies have also highlighted the growing importance of intelligent analytics, cloud-assisted computing, and distributed workload management in supporting scalable software-intensive environments, suggesting additional opportunities for extending and evaluating integrated software development frameworks [24, 25].

Future research should validate the framework through implementation in industrial or academic environments using objective software engineering metrics such as deployment frequency, LTC, defect density, service coupling, CRR, and system availability. Such implementation-based evaluation would enable quantitative comparison between standalone XP, DevOps, MSA, and the proposed integrated framework, thereby providing stronger empirical evidence of its technical effectiveness.

5. CONCLUSION

This study investigated the feasibility of integrating XP–DevOps–MSA into a unified software development framework. The results indicate strong practitioner agreement regarding the applicability of the proposed approach, particularly in terms of scalability, component reusability, deployment readiness, development efficiency, and software quality. Participants also identified challenges related to architectural complexity and the specialized skills required to manage distributed microservices and automated delivery pipelines.

The study contributes to software engineering research by proposing a structured XP–DevOps–MSA framework and providing empirical evidence of its feasibility through practitioner-based evaluation. In addition, a technical evaluation framework based on objective software engineering metrics, including CRR, deployment frequency, and LTC, is presented to support future implementation-based validation.

Future work should focus on implementing the framework in industrial or academic environments and evaluating its effectiveness using objective performance metrics. Longitudinal and multi-domain case studies would further

strengthen the generalizability and practical applicability of the proposed framework.

REFERENCES

[1] Dybå, T., Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. Information and Software Technology, 50(9-10): 833-859. https://doi.org/10.1016/j.infsof.2008.01.006

[2] Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B. (2012). A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software, 85(6): 1213-1221. https://doi.org/10.1016/j.jss.2012.02.033

[3] Di Francesco, P., Lago, P., Malavolta, I. (2018). Migrating towards microservice architectures: An industrial survey. In 2018 IEEE international conference on software architecture (ICSA), Seattle, WA, USA, pp. 29-2909, https://doi.org/10.1109/ICSA.2018.00012

[4] Jamshidi, P., Pahl, C., Mendonça, N.C., Lewis, J., Tilkov, S. (2018). Microservices: The journey so far and challenges ahead. IEEE Software, 35(3): 24-35. https://doi.org/10.1109/MS.2018.2141039

[5] Ünlü, H., Kennouche, D.E., Soylu, G.K., Demirörs, O. (2024). Microservice-based projects in agile world: A structured interview. Information and Software Technology, 165: 107334. https://doi.org/10.1016/j.infsof.2023.107334

[6] Moreschini, S., Pour, S., Lanese, I., Balouek, D., Bogner, J., Li, X., Pecorelli, F., Soldani, J., Truyen, E., Taibi, D. (2025). AI techniques in the microservices life-cycle: A systematic mapping study. Computing, 107: 100. https://doi.org/10.1007/s00607-025-01432-z

[7] Balalaie, A., Heydarnoori, A., Jamshidi, P. (2016). Microservices architecture enables DevOps: Migration to a cloud-native architecture. IEEE Software, 33(3): 42-52. https://doi.org/10.1109/MS.2016.64

[8] Alshuqayran, N., Ali, N., Evans, R. (2016). A systematic mapping study in microservice architecture. In 2016 IEEE 9th International Conference on Service-Oriented Computing and Applications (SOCA), Macau, China, pp. 44-51, https://doi.org/10.1109/SOCA.2016.15.

[9] Khoso, F.H., Lakhan, A., Arain, A.A., Soomro, M.A., Nizamani, S.Z., Kanwar, K. (2021). A microservice-based system for industrial Internet of Things in fog-cloud assisted network. Engineering, Technology & Applied Science Research, 11(2): 7029-7032. https://doi.org/10.48084/etasr.4077

[10] Lwakatare, L.E., Kilamo, T., Karvonen, T., Sauvola, T., Heikkilä, V., Itkonen, J., Kuvaja, P., Mikkonen, T., Oivo, M., Lassenius, C. (2019). DevOps in practice: A multiple case study of five companies. Information and Software Technology, 114: 217-230. https://doi.org/10.1016/j.infsof.2019.06.010

[11] Waseem, M., Liang, P., Shahin, M. (2020). A systematic mapping study on microservices architecture in DevOps. Journal of Systems and Software, 170: 110798. https://doi.org/10.1016/j.jss.2020.110798

[12] Shahin, M., Babar, M.A., Zhu, L.M. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. IEEE Access, 5: 3909-3943. https://doi.org/10.1109/ACCESS.2017.2685629

[13] Ayas, H.M., Leitner, P., Hebig, R. (2023). An empirical study of the systemic and technical migration towards microservices. Empirical Software Engineering, 28: 85. https://doi.org/10.1007/s10664-023-10308-9

[14] Binamungu, L.P., Maro, S. (2023). Behaviour driven development: A systematic mapping study. Journal of Systems and Software, 203: 111749. https://doi.org/10.1016/j.jss.2023.111749

[15] Tuglular, T., Coşkun, D.E., Gülen, Ö., Okluoğlu, A., Algan, K. (2021). Behavior-driven development of microservice applications. International Journal of Computers, 15: 130-137. https://doi.org/10.46300/9108.2021.15.20

[16] Sabuhi, M., Musilek, P., Bezemer, C.P. (2024). Micro-FL: A fault-tolerant scalable microservice-based platform for federated learning. Future Internet, 16(3): 70. https://doi.org/10.3390/fi16030070

[17] Soldani, J., Tamburri, D.A., Van Den Heuvel, W.J. (2018). The pains and gains of microservices: A systematic grey literature review. Journal of Systems and Software, 146: 215-232. https://doi.org/10.1016/j.jss.2018.09.082

[18] Ait Said, M., Ezzati, A., Mihi, S., Belouaddane, L. (2024). Microservices adoption: An industrial inquiry into factors influencing decisions and implementation strategies. International Journal of Computing and Digital Systems, 15(1): 1417-1432. https://doi.org/10.12785/ijcds/1501100

[19] Ortiz, G., Boubeta-Puig, J., Criado, J., Corral-Plaza, D., Garcia-de-Prado, A., Medina-Bulo, I., Iribarne, L. (2022). A microservice architecture for real-time IoT data processing: A reusable Web of Things approach for smart ports. Computer Standards & Interfaces, 81: 103604. https://doi.org/10.1016/j.csi.2021.103604

[20] Erich, F., Amrit, C., Daneva, M. (2017). A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process, 29(6): e1885. https://doi.org/10.1002/smr.1885

[21] Pahl, C., Jamshidi, P., Zimmermann, O. (2020). Architectural principles for cloud software. ACM Transactions on Internet Technology, 18(2): 1-23. https://doi.org/10.1145/3104028

[22] Frakes, W., Terry, C. (1996). Software reuse: Metrics and models. ACM Computing Surveys, 28(2): 415-435. https://doi.org/10.1145/234528.234531

[23] Erich, F.M.A., Amrit, C., Daneva, M. (2017). A qualitative study of DevOps usage in practice. Journal of Software: Evolution and Process, 29(6): e1885. https://doi.org/10.1002/smr.1885

[24] Puttaswamy, N.G., Murthy, A.N., Degha, H. (2024). A comparative review of internet of things model workload distribution techniques in fog computing networks. Information Dynamics and Applications, 3(1): 21-46. https://doi.org/10.56578/ida030103

[25] Baig, M.D., Akram, W., Ul Haq, H.B., Rajput, H.Z., Imran, M. (2024). Optimizing misinformation control: A cloud-enhanced machine learning approach. Information Dynamics and Applications, 3(1): 1-11. https://doi.org/10.56578/ida030101