Socio-Technical Design, Organizational Structures, and Collaboration Dynamics in Software Development Organizations: A Qualitative Systematic Literature Review

Maradoni Louisse A. Ambrad[1], Stephanie Dane S. Salvador[2], Russel M. Dela Torre[3]

[1]North Negros College, Cadiz City, Philippines

[2]Department of Education, Sagay City Philippines

[3]Carlos Hilado Memorial State University, Talisay City, Philippines

DOI: https://doi.org/10.47772/IJRISS.2026.100500688

Received: 24 May 2026; Accepted: 29 May 2026; Published: 11 June 2026

ABSTRACT

Software development organizations are reshaping their structures in response to Agile transformation, DevOps adoption, and distributed collaboration, but empirical evidence on how organizational and socio-technical configurations jointly shape collaboration and delivery performance remains fragmented across disciplines. Although numerous studies have independently examined software engineering teams, Agile governance, and collaboration mechanisms, evidence regarding organizational and socio-technical dynamics remains fragmented across software engineering and information systems literature. This systematic literature review synthesized evidence concerning organizational structures and socio-technical factors influencing collaboration and coordination in software development organizations.

The review followed the PRISMA 2020 reporting framework and adopted the Kitchenham and Charters Guidelines for evidence-based software engineering systematic literature reviews. Using a predefined search string and PICOC framework, 353 studies were initially identified through database and registry searching procedures. Following filtering, eligibility screening, and quality assessment conducted independently by three reviewers, 21 high-quality studies published between 2022 and 2026 were retained for final synthesis.

The findings revealed that organizational structures significantly influence software engineering outcomes through governance mechanisms, team autonomy, architecture-team alignment, communication structures, and knowledge mobilization practices. Agile and DevOps-oriented structures demonstrated stronger coordination efficiency, communication flexibility, and software delivery effectiveness compared to rigid hierarchical structures. The review also identified major socio-technical factors affecting collaboration and coordination, including communication systems, psychological safety, organizational culture, human factors, distributed coordination mechanisms, and socio-technical alignment between technical systems and organizational workflows.

This qualitative systematic literature review integrates recent evidence on organizational structures and socio-technical factors that influence collaboration and coordination in software development organizations.

Keywords: software engineering, socio-technical systems, Agile Development, DevOps, collaboration coordination

INTRODUCTION

Software development organizations continue to evolve rapidly due to increasing technological complexity, digital transformation, distributed collaboration, and the widespread adoption of Agile and DevOps methodologies. Contemporary software engineering environments now operate within highly interconnected ecosystems where organizational structures, communication systems, coordination mechanisms, and socio-

technical interaction significantly influence software delivery performance and project outcomes. As organizations adopt large-scale Agile practices, cloud-native architectures, microservices, distributed development, and continuous integration and deployment models, software engineering teams increasingly depend on effective organizational and socio-technical integration to sustain productivity, innovation, and collaboration.

Traditional hierarchical software engineering structures have gradually transitioned toward more adaptive organizational arrangements emphasizing cross-functional collaboration, iterative development, distributed coordination, and team autonomy. Frameworks such as Scrum, DevOps, and the Scaled Agile Framework (SAFe) have reshaped organizational governance and interaction patterns within software development organizations. These organizational transformations aim to improve responsiveness, software quality, and delivery efficiency. However, they also introduce new coordination challenges involving governance alignment, communication fragmentation, socio-technical dependencies, and knowledge mobilization.

Organizational structures strongly influence how software engineering teams coordinate tasks, communicate across technical boundaries, share knowledge, and respond to changing project requirements. Studies examining Agile governance, architecture-team alignment, distributed collaboration, and DevOps coordination suggest that organizational arrangements directly affect software quality, team effectiveness, project success, and productivity outcomes. Nevertheless, evidence regarding the influence of organizational structures on software engineering performance remains dispersed across empirical software engineering and organizational studies.

In parallel, socio-technical systems theory has become increasingly important in understanding software engineering collaboration and coordination. Software development organizations function not only through technical infrastructures but also through social interaction, psychological safety, communication quality, organizational culture, human factors, and collaborative work practices. Socio-technical factors such as collaboration platforms, distributed coordination tools, knowledge-sharing systems, and communication mechanisms significantly affect coordination efficiency and communication effectiveness in software development environments.

Despite the growing body of literature examining software engineering collaboration, existing evidence remains fragmented between organizational and socio-technical perspectives. Many studies focus narrowly on Agile methodologies, DevOps implementation, or collaboration technologies without consolidating broader organizational and socio-technical dynamics affecting software engineering teams. Consequently, there remains limited synthesis integrating organizational structures and socio-technical systems within contemporary software development environments.

Existing literature reviews have largely examined Agile governance, DevOps adoption, distributed software development, or collaboration technologies independently. However, limited systematic evidence synthesis has integrated organizational structures and socio-technical collaboration dynamics within a unified software engineering perspective. As contemporary software development environments increasingly rely on cross-functional coordination, distributed collaboration, and adaptive governance structures, there remains a need for a comprehensive synthesis examining how organizational configurations and socio-technical systems collectively influence collaboration, coordination, and software engineering outcomes. This review addresses this gap by consolidating contemporary evidence concerning organizational structures and socio-technical collaboration mechanisms within software development organizations.

This systematic literature review addresses this research gap by synthesizing evidence concerning organizational structures and socio-technical factors influencing collaboration and coordination in software development organizations. The review follows the PRISMA 2020 reporting framework and adopts the Kitchenham and Charters Guidelines for software engineering systematic literature reviews.

The study was guided by the following research questions:

RQ1: How do different organizational structures influence processes and outcomes in software development teams?

RQ2: What socio-technical factors influence collaboration and coordination in software development teams?

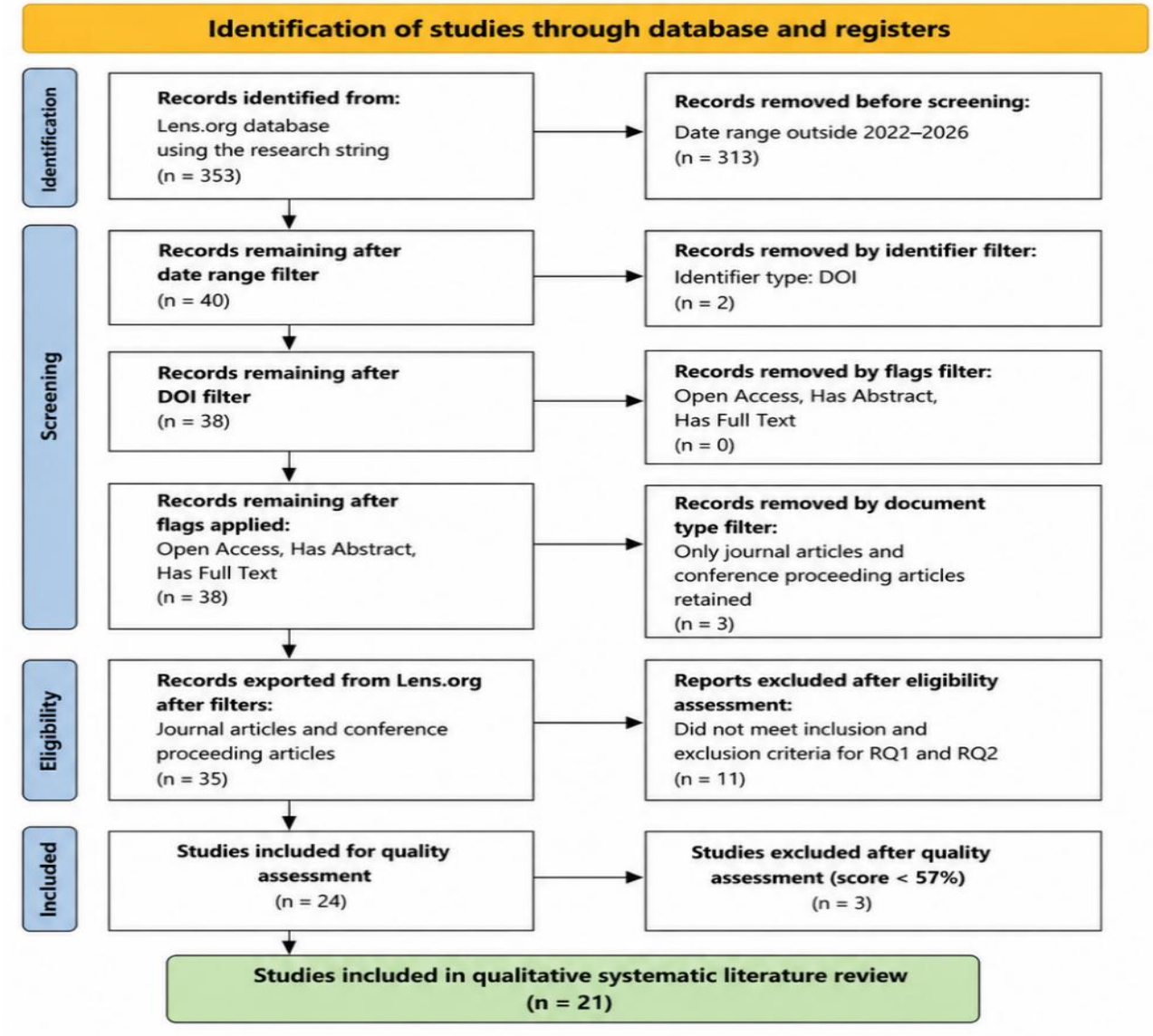
The review contributes to software engineering research by consolidating evidence concerning organizational structures, socio-technical systems, communication mechanisms, collaboration dynamics, and software engineering outcomes. The findings provide implications for Agile governance, DevOps coordination, distributed collaboration, software engineering management, and future socio-technical software engineering research.

METHODS

Research Design

This study employed a qualitative systematic literature review design following the PRISMA 2020 reporting framework and the evidence-based software engineering guidelines proposed by Kitchenham and Charters. The systematic review approach was selected to ensure methodological rigor, transparency, reproducibility, and comprehensive evidence synthesis across diverse software engineering studies examining organizational structures and socio-technical collaboration. The review process consisted of systematic searching, eligibility screening, quality assessment, data extraction, and qualitative thematic synthesis. The methodology emphasized evidence-based software engineering principles to strengthen the validity and reliability of the review findings.

Figure 1. PRISMA 2020 Framework Model



Formulation of Research Questions

The formulation of the research questions was guided by the growing complexity of contemporary software development environments and the increasing importance of organizational and socio-technical dynamics in software engineering practice. Modern software engineering organizations operate within Agile, DevOps, distributed, and large-scale collaborative environments where organizational structures, communication systems, coordination mechanisms, and human–technology interaction significantly influence software delivery outcomes. Existing literature demonstrates that software development success is not solely dependent on technical capability but also on organizational governance, collaboration efficiency, communication quality, and socio-technical alignment.

Preliminary exploration of the literature revealed that many studies independently discussed organizational structures, Agile governance, team autonomy, DevOps coordination, distributed collaboration, communication systems, psychological safety, and knowledge-sharing practices. However, the findings remained fragmented across software engineering, organizational theory, information systems, and socio-technical systems literature. Furthermore, there was limited evidence synthesis integrating organizational structures and socio-technical factors within a single systematic review focused on software development organizations.

The formulation of the research questions was therefore intended to address two major dimensions emerging from the preliminary literature exploration. The first dimension focused on organizational structures and their influence on software engineering processes and outcomes. The second dimension focused on socio-technical factors influencing collaboration and coordination within software development teams. These dimensions were aligned with the principles of socio-technical systems theory and evidence-based software engineering research.

Consequently, the study formulated the following research questions:

RQ1: How do different organizational structures influence processes and outcomes in software development teams?

RQ2: What socio-technical factors influence collaboration and coordination in software development teams?

The research questions were further refined using the PICOC framework to ensure alignment between the objectives of the study, the search strategy, the inclusion and exclusion criteria, and the synthesis procedures. The questions were intentionally designed to capture both organizational and socio-technical perspectives influencing software engineering collaboration, communication, coordination, and project outcomes.

Table 1. PICOC Framework for RQ1

PICOC Element	Specification
Population	Software development teams and software engineering organizations
Intervention	Organizational structures including hierarchical, Agile, matrix, and flat organizational models
Comparison	Different organizational structures such as hierarchical versus Agile arrangements
Outcome	Team performance, software quality, productivity, coordination efficiency, and project success
Context	Agile, DevOps, and technology-driven software development environments

Table 1 presents the PICOC framework used to guide the first research question, which examined how organizational structures influence software development processes and outcomes. The framework clarified the analytical boundaries of the review by identifying software development teams and software engineering

organizations as the primary population of interest. The intervention focused on organizational structures such as hierarchical, Agile, matrix, and flat organizational models commonly used in software engineering environments. The comparison element allowed the study to examine differences between structural arrangements, particularly centralized versus Agile and adaptive team configurations. The selected outcomes emphasized software engineering performance indicators including software quality, productivity, coordination efficiency, team performance, and project success. The context emphasized Agile, DevOps, and technology-driven software development environments where organizational structure strongly influences communication, governance, and technical coordination. Overall, the framework ensured that the literature search, screening, and synthesis remained aligned with the organizational dimension of the review.

Table 2. PICOC Framework for RQ2

PICOC Element	Specification
Population	Software engineering teams
Intervention	Socio-technical factors including collaboration tools, communication systems, human–technology interaction, and coordination mechanisms
Comparison	Effective versus ineffective socio-technical alignment
Outcome	Collaboration quality, coordination effectiveness, communication efficiency, and organizational collaboration outcomes
Context	Digital and technology-driven software development environments

Table 2 presents the PICOC framework developed for the second research question, which focused on socio-technical factors affecting collaboration and coordination in software development teams. The framework identified software engineering teams as the population because collaboration and coordination processes primarily occur within team-based software development environments. The intervention focused on socio-technical factors such as collaboration platforms, communication systems, coordination mechanisms, and human–technology interaction that influence software engineering practices. The comparison component examined effective versus ineffective socio-technical alignment to determine how different organizational and technical conditions affect collaborative performance. The outcomes emphasized collaboration quality, communication efficiency, coordination effectiveness, and broader organizational collaboration outcomes. The context highlighted digital and technology-driven software engineering environments where distributed work, Agile practices, DevOps processes, and collaborative technologies shape team interaction. The framework strengthened the consistency of the review by ensuring that the search strategy, inclusion criteria, and synthesis procedures remained directly connected to socio-technical collaboration in software development organizations.

Search Strategy

To identify relevant studies addressing the research questions, a systematic search strategy was developed using keywords associated with software engineering teams, organizational structures, communication systems, coordination mechanisms, and organizational learning. The search string was constructed using Boolean operators to maximize retrieval of studies related to organizational and socio-technical dynamics in software development environments.

The following search string was used during the initial literature identification and screening process:

(“software development teams” OR “software engineering teams”) AND (“organizational structure” OR “organizational design” OR “structural configuration” OR “team structure”) AND (“decision-making” OR “coordination mechanisms” OR “communication structure” OR “organizational learning”)

The search string was designed to capture studies discussing organizational structures, communication systems, collaboration mechanisms, socio-technical interaction, governance arrangements, and software engineering coordination processes. The use of Boolean operators and alternative terminology improved the comprehensiveness of the search strategy and ensured broader coverage of relevant software engineering and organizational studies.

The systematic search process was conducted using Lens.org as the primary scholarly indexing platform. Lens.org was selected because it aggregates records from multiple scholarly repositories including CrossRef, OpenAlex, PubMed, and publisher databases, enabling broader interdisciplinary retrieval across software engineering and information systems literature. The search strategy followed the PICOC framework and used Boolean operators to maximize retrieval sensitivity while maintaining relevance to organizational and socio-technical software engineering studies.

To support transparent and systematic screening procedures, the retrieved studies were exported and organized using Rayyan systematic review software. Rayyan facilitated blinded title and abstract screening, duplicate checking, eligibility assessment, and reviewer conflict resolution. The screening process was independently conducted by three reviewers to strengthen methodological reliability and minimize selection bias.

The review intentionally focused on studies published between 2022 and 2026 to capture contemporary evidence reflecting recent developments in Agile transformation, DevOps adoption, distributed collaboration, cloud-native software engineering, and large-scale organizational coordination practices. The review focused on publications from 2022 to 2026, written in English, with available full text, and published as scholarly journal articles or conference papers. Although the exported records came from a specific database source, the review used PRISMA logic to make the screening and inclusion process transparent.

Inclusion and Exclusion Criteria

The inclusion criteria required studies to focus on software development organizations, software engineering teams, organizational structures, socio-technical systems, communication mechanisms, collaboration processes, or software engineering outcomes. Eligible studies were required to be peer-reviewed journal articles or conference proceedings published between 2022 and 2026 with accessible full text and sufficient methodological detail.

Studies unrelated to software engineering contexts, lacking organizational or socio-technical relevance, demonstrating insufficient methodological clarity, or failing to address collaboration or coordination outcomes were excluded from the review.

Table 3. Eligibility Criteria

Category	Inclusion Criteria	Exclusion Criteria
Context	Studies focusing on software development teams, software engineering organizations, or IT project environments	Studies not related to software development or IT contexts
Organizational relevance	Studies addressing organizational structures, organizational design, or organizational theory	Studies with no organizational or theoretical focus
Socio-technical relevance	Studies examining socio-technical factors or socio-technical systems theory	Studies that do not consider human–technology interaction or socio-technical aspects
Knowledge processes	Studies discussing knowledge management, creation, sharing, transfer, or integration	Studies that do not address knowledge-related processes

Category	Inclusion Criteria	Exclusion Criteria
Challenges and limitations	Studies identifying organizational challenges or design limitations	Studies not reporting challenges, barriers, or limitations
Outcomes	Studies reporting performance, collaboration, coordination, communication, knowledge sharing, software quality, or project success	Studies lacking clear outcomes or findings related to team or organizational effectiveness
Study type	Empirical studies and systematic reviews	Opinion papers, blogs, editorials, and non-peer-reviewed sources
Publication quality	Peer-reviewed journal articles and conference papers	Non-scholarly publications
Time frame	Studies published between 2022 and 2026	Studies published outside the defined time frame
Language and access	English-language studies with full text available	Non-English, abstract-only, or inaccessible studies
Uniqueness	Unique studies	Duplicate publications

Table 3 shows that the eligibility criteria were intentionally broad enough to capture studies relevant to both research questions while also narrow enough to exclude literature that lacked software engineering, organizational, socio-technical, or outcome relevance. The criteria also explain why some studies that addressed general teamwork, healthcare, urban systems, or broad digital transformation were not retained unless they had direct software development relevance.

Screening and Selection Process

The screening and selection process followed the PRISMA-based systematic review procedure to ensure a transparent and systematic identification of relevant studies. The initial database and registry search produced 353 records related to software development organizations, organizational structures, Agile practices, DevOps environments, collaboration systems, and socio-technical factors. Publication year filters were applied to retain contemporary studies, reducing the records to 40 studies. DOI filtering further reduced the records to 38 studies, while document type restrictions retained only peer-reviewed journal articles and conference papers, resulting in 35 studies eligible for screening.

The remaining studies underwent title, abstract, and full-text evaluation based on the predefined inclusion and exclusion criteria aligned with the PICOC framework and research questions. Following eligibility assessment and quality evaluation procedures, 24 studies were retained for the final qualitative synthesis because they demonstrated sufficient methodological rigor and direct relevance to the objectives of the review.

Table 4. PRISMA Screening Summary

PRISMA Stage	Number of Records
Records identified through database searching	353
Records after date filtering	40
Records after DOI filtering	38
Records after document filtering	35

PRISMA Stage	Number of Records
Records screened by title and abstract	35
Records excluded after screening	11
Studies assessed for eligibility	24
Studies included after quality assessment	21

Table 4 summarizes the PRISMA-based screening and selection process. The initial search identified 353 studies through database and registry searching procedures. Sequential filtering based on publication year, DOI availability, document type restrictions, title and abstract relevance, eligibility criteria, and quality assessment procedures reduced the evidence base to 21 high-quality studies included in the final qualitative synthesis.

Quality Assessment

The quality assessment process followed the Kitchenham and Charters Guidelines for evidence-based software engineering systematic literature reviews. The assessment evaluated methodological rigor, organizational relevance, socio-technical contribution, empirical grounding, and validity of conclusions. Seven predefined quality assessment criteria were used throughout the evaluation process.

Table 5. Quality Assessment Criteria

Code	Quality Assessment Question
QA1	Was the research objective clearly focused on software development organizations, organizational structures, or socio-technical collaboration?
QA2	Was the study design appropriate for addressing the research objectives?
QA3	Were the data collection and analysis procedures clearly explained and methodologically sound?
QA4	Did the study provide findings related to collaboration, coordination, communication, software quality, productivity, or project success?
QA5	Were organizational structures, socio-technical factors, or knowledge processes clearly identified and discussed?
QA6	Were organizational challenges, barriers, or limitations adequately identified and analyzed?
QA7	Were the conclusions valid, evidence-based, and relevant to software engineering organizations?

The quality assessment was independently conducted by three reviewers to improve inter-review reliability and minimize evaluator bias. Each reviewer evaluated all initially included studies using the predefined criteria.

Table 6. Quality Assessment Results

Reviewer	Number of Papers	Studies Above 57% Threshold	Percentage
Reviewer 1	24	22	91.7%
Reviewer 2	24	22	91.7%
Reviewer 3	24	21	87.5%

The quality assessment results demonstrated strong methodological consistency across reviewers. Following consensus evaluation, studies consistently demonstrating weaker methodological quality and limited alignment

with RQ1 and RQ2 were excluded from the final synthesis. Consequently, the final evidence base consisted of 21 high-quality studies.

Data Synthesis

The review employed qualitative thematic synthesis and narrative analysis procedures. Extracted findings were coded and categorized according to the two research questions. Themes associated with organizational structures and software engineering outcomes were synthesized under RQ1, while socio-technical collaboration factors were synthesized under RQ2.

The synthesis process emphasized identifying recurring organizational patterns, coordination mechanisms, collaboration dynamics, communication structures, and socio-technical interactions across the included studies.

RESULTS

The systematic review process resulted in the inclusion of 21 high-quality studies examining organizational structures, socio-technical systems, communication mechanisms, Agile governance, DevOps coordination, distributed collaboration, and software engineering outcomes within technology-driven software development environments. The included studies represented methodological diversity including ethnographic investigations, empirical studies, systematic literature reviews, organizational analyses, qualitative investigations, and conceptual-empirical research.

Following independent reviewer evaluation using the predefined quality assessment criteria, studies demonstrating weaker methodological rigor, insufficient software engineering relevance, or limited alignment with the research questions were excluded from the final synthesis. The retained studies provided a sufficiently diverse and methodologically reliable evidence base for answering RQ1 and RQ2.

Table 7. Final Included Studies After Quality Assessment and Reviewer Consolidation

Study	Methodology	Primary Focus	Alignment	Reviewer Notes	Inclusion Decision
Zhou et al. (2022)	Ethnographic study	DevOps and microservices organizational structures	RQ1, RQ2	Strong empirical grounding and comprehensive socio-technical analysis	Included
Gustavsson et al. (2022)	Multiple case study	SAFe implementation and team autonomy	RQ1	Strong governance and coordination findings with high methodological rigor	Included
Dingsøyr et al. (2024)	Empirical study	Teamwork quality in large-scale Agile projects	RQ1, RQ2	Strong collaboration and project success analysis	Included
Moser & Moura (2022)	Systematic literature review	People maturity models in software teams	RQ1	Strong organizational maturity and collaboration discussion	Included
Klymenko et al. (2022)	Qualitative organizational study	Privacy compliance and technical coordination	RQ2	Strong socio-technical governance alignment	Included
Nazir et al. (2022)	Empirical case study	Hyper-Agile emergency software environments	RQ1, RQ2	Strong adaptive coordination and	Included

Study	Methodology	Primary Focus	Alignment	Reviewer Notes	Inclusion Decision
				organizational flexibility discussion	
Trzeciak et al. (2022)	Organizational innovation study	Open innovation in software micro-organizations	RQ1	Moderate empirical depth but strong organizational relevance	Included
Abid-Baudin (2022)	Organizational analysis	Knowledge-intensive software organizations	RQ1	Strong organizational contribution with moderate empirical detail	Included
Iffah et al. (2023)	Empirical organizational study	Organizational and process success factors	RQ1	Clear discussion of software development success variables	Included
Grossman et al. (2024)	Conceptual-empirical investigation	Fluid team effectiveness	RQ2	Strong collaboration framework contribution	Included
Pokhrel & Goyal (2022)	Survey-based study	Psychological workplace climate and well-being	RQ2	Strong human factors and communication discussion	Included
Edison et al. (2022)	Systematic literature review	Large-scale Agile methodologies	RQ1	Comprehensive Agile governance synthesis with strong methodological rigor	Included
Ijaz et al. (2022)	Systematic review	Distributed Agile task allocation	RQ2	Strong distributed coordination and communication analysis	Included
Hustad et al. (2022)	Systematic literature review	Knowledge mobilization in Agile projects	RQ2	Strong knowledge-sharing and organizational learning synthesis	Included
Olivares et al. (2024)	Intelligent systems study	Agile team allocation optimization	RQ1, RQ2	Strong coordination optimization and collaboration analysis	Included
Tamburri et al. (2023)	Empirical investigation	Software architecture and team organization	RQ1	Strong architecture-team alignment and DevOps coordination findings	Included
Buvik & Tkalich (2022)	Empirical study	Psychological safety in Agile software teams	RQ2	Strong collaboration and trust-related findings	Included
Kalluri (2022)	Human factors study	Agile Scrum risk management	RQ2	Strong human-centered coordination and risk management analysis	Included
Almeida & Espinheira (2022)	Organizational governance study	Agile governance and coordination	RQ1	Strong large-scale Agile coordination discussion	Included

Study	Methodology	Primary Focus	Alignment	Reviewer Notes	Inclusion Decision
Matthews & Tanner (2022)	Cultural collaboration analysis	National culture and Agile collaboration	RQ2	Strong communication and cultural coordination findings	Included
Kukhareva et al. (2022)	Organizational communication study	Coordination and communication structures	RQ1, RQ2	Strong socio-technical collaboration relevance	Included

The final evidence base demonstrates that contemporary software engineering research strongly emphasizes Agile governance, DevOps coordination, organizational communication, socio-technical collaboration, distributed software engineering, and knowledge-sharing mechanisms. The methodological diversity of the retained studies further strengthened the robustness of the synthesis by integrating evidence from empirical software engineering investigations, systematic literature reviews, organizational analyses, qualitative studies, and collaboration-centered research.

RQ1: Organizational Structures and Software Development Outcomes

The findings revealed that organizational structures significantly influence software engineering performance, communication effectiveness, governance alignment, collaboration quality, and project success. Agile-oriented organizational arrangements consistently demonstrated stronger adaptability, coordination responsiveness, and communication flexibility compared to traditional hierarchical structures.

Several studies emphasized the importance of governance mechanisms and team autonomy within large-scale Agile environments. Gustavsson et al. (2022) demonstrated that SAFe implementations reshaped team autonomy through structured governance and coordination systems balancing organizational control with Agile flexibility. Similarly, Edison et al. (2022) identified governance alignment, coordination frameworks, and communication structures as major determinants of large-scale Agile effectiveness.

DevOps-oriented organizational structures also significantly influenced software engineering outcomes. Zhou et al. (2022) demonstrated that architecture-team alignment and cross-functional DevOps collaboration improved communication efficiency and reduced organizational fragmentation. Tamburri et al. (2023) similarly identified strong relationships between software architecture structures, communication systems, and coordination effectiveness within DevOps environments.

Organizational maturity and knowledge management structures were also strongly associated with software engineering performance. Moser and Moura (2022) emphasized the importance of people-management maturity in improving productivity, collaboration, and organizational learning within software development teams. Hustad et al. (2022) further demonstrated that knowledge mobilization structures significantly improved collaboration and adaptability within Agile information systems projects.

The review also identified several organizational barriers affecting collaboration and software engineering outcomes. Hierarchical rigidity, fragmented communication structures, weak governance alignment, and inadequate coordination mechanisms were repeatedly associated with collaboration inefficiencies and reduced organizational adaptability.

Table 8. Organizational Structure Themes and Outcomes

Study	Organizational Focus	Key Findings	Outcomes
Gustavsson et al. (2022)	SAFe and team autonomy	Governance structures influenced autonomy and coordination	Improved collaboration and coordination efficiency

Study	Organizational Focus	Key Findings	Outcomes
Edison et al. (2022)	Large-scale Agile governance	Agile governance improved organizational responsiveness	Higher software delivery effectiveness
Zhou et al. (2022)	DevOps organizational alignment	Architecture-team congruence improved coordination	Reduced communication fragmentation
Tamburri et al. (2023)	DevOps team organization	Organizational alignment enhanced collaboration	Improved software architecture coordination
Moser & Moura (2022)	People management maturity	Organizational maturity improved teamwork quality	Increased productivity and collaboration
Hustad et al. (2022)	Knowledge mobilization	Knowledge-sharing systems enhanced organizational learning	Improved Agile adaptability

The findings collectively demonstrate that adaptive organizational structures emphasizing governance alignment, cross-functional collaboration, and communication flexibility positively influence software engineering performance and organizational effectiveness.

RQ2: Socio-Technical Factors Influencing Collaboration and Coordination

The findings revealed that socio-technical factors strongly influence collaboration quality, communication effectiveness, coordination efficiency, and software engineering outcomes. Communication systems emerged as one of the strongest determinants of successful software engineering collaboration.

Psychological safety was consistently identified as a critical contributor to Agile collaboration and team performance. Buvik and Tkalich (2022) demonstrated that psychologically safe environments improved communication openness, collaborative problem-solving, and coordination effectiveness in Agile software development teams.

Human factors and organizational culture also strongly influenced socio-technical collaboration. Matthews and Tanner (2022) identified national cultural differences as major influences on Agile communication practices and collaboration expectations. Similarly, Kalluri (2022) emphasized the role of human factors and risk management in sustaining coordination within complex Agile Scrum environments.

Distributed collaboration introduced additional socio-technical coordination challenges. Ijaz et al. (2022) demonstrated that distributed Agile environments required strong task allocation systems, communication transparency, and coordination mechanisms to sustain collaboration across geographically dispersed teams.

Knowledge-sharing mechanisms emerged as another major socio-technical factor affecting collaboration quality. Hustad et al. (2022) demonstrated that organizational knowledge mobilization improved collaborative learning, coordination effectiveness, and project adaptability.

Privacy engineering and governance studies further emphasized the importance of socio-technical alignment between technical infrastructures and organizational processes. Klymenko et al. (2022) demonstrated that effective privacy compliance depended heavily on coordination between technical implementation and organizational interpretation.

Table 9. Socio-Technical Factors Influencing Collaboration and Coordination

Study	Socio-Technical Focus	Key Findings	Collaboration Impact
Buvik & Tkalich (2022)	Psychological safety	Safe environments improved communication openness	Stronger team collaboration

Study	Socio-Technical Focus	Key Findings	Collaboration Impact
Matthews & Tanner (2022)	Organizational culture	Cultural factors influenced Agile communication	Improved coordination awareness
Ijaz et al. (2022)	Distributed Agile coordination	Task allocation systems improved distributed teamwork	Enhanced coordination efficiency
Hustad et al. (2022)	Knowledge mobilization	Knowledge-sharing mechanisms strengthened learning	Improved collaborative adaptability
Klymenko et al. (2022)	Privacy engineering collaboration	Technical-organizational alignment improved compliance coordination	Enhanced socio-technical integration
Kalluri (2022)	Human factors and Agile risks	Human-centered coordination reduced project risks	Improved collaboration stability

The findings indicate that successful software engineering collaboration depends on strong socio-technical alignment involving communication systems, organizational culture, human factors, coordination structures, and knowledge-sharing practices.

DISCUSSION

The review demonstrates that organizational structures and socio-technical systems are deeply interconnected within software engineering organizations. Agile and DevOps structures improve software engineering performance not only through process adaptation but also through socio-technical alignment enabling communication flexibility, coordination responsiveness, and collaborative integration.

The synthesis further indicates that organizational effectiveness in software development environments depends heavily on balancing governance structures with team autonomy. Excessive organizational rigidity may reduce collaboration flexibility and innovation capacity, whereas insufficient coordination structures may create communication fragmentation and project instability.

The findings also reinforce socio-technical systems theory within software engineering contexts. Technical infrastructures alone are insufficient to sustain collaboration and coordination effectiveness. Instead, collaboration quality depends heavily on social interaction, communication transparency, psychological safety, organizational culture, and knowledge-sharing systems.

The review further demonstrates that distributed software engineering environments intensify the importance of socio-technical coordination. Distributed Agile development, DevOps collaboration, and large-scale software engineering environments require stronger communication systems, task allocation mechanisms, and organizational coordination frameworks to sustain effective collaboration.

Although Agile and DevOps-oriented organizational structures generally demonstrated positive effects on collaboration and coordination, the findings also suggest emerging tensions between governance standardization and team autonomy within large-scale software engineering environments. Several studies implied that highly structured Agile governance mechanisms may unintentionally constrain flexibility and increase coordination complexity when implemented across enterprise-scale environments. Similarly, while DevOps structures reduced communication silos and improved cross-functional integration, they also introduced increased cognitive and coordination demands on software engineering teams. Distributed collaboration environments further intensified socio-technical challenges associated with communication transparency, tacit knowledge exchange, and organizational cohesion. These findings indicate that organizational adaptation alone may not guarantee software engineering effectiveness unless accompanied by sustained socio-technical alignment and human-centered coordination practices.

These findings have important implications for software engineering management. Organizations implementing

Agile, DevOps, or distributed collaboration models should invest not only in technical infrastructures but also in communication systems, knowledge-sharing mechanisms, psychologically safe environments, and organizational coordination practices.

LIMITATIONS

This systematic literature review has several limitations. First, the review primarily relied on Lens.org as the indexing platform, which may have limited retrieval coverage despite its aggregation of multiple scholarly repositories. Second, the review included only English-language studies published between 2022 and 2026, potentially excluding relevant earlier or non-English literature. Third, the review focused exclusively on peer-reviewed journal articles and conference proceedings, thereby excluding gray literature, industrial reports, and practitioner publications that may contain additional organizational insights. Fourth, although the review employed independent reviewer assessment procedures, qualitative synthesis and thematic interpretation may still contain elements of subjective interpretation. Finally, the rapid evolution of Agile, DevOps, distributed collaboration, and socio-technical software engineering practices may result in emerging evidence beyond the temporal scope of the review.

CONCLUSION

This systematic literature review synthesized evidence concerning organizational structures and socio-technical factors influencing collaboration and coordination in software development organizations. Following the PRISMA 2020 framework and the Kitchenham and Charters Guidelines, the review consolidated evidence from 21 high-quality studies published between 2022 and 2026.

The findings confirmed that organizational structures significantly influence software engineering outcomes through governance mechanisms, architecture-team alignment, communication systems, team autonomy, and knowledge management practices. Agile and DevOps-oriented structures demonstrated stronger coordination efficiency, communication flexibility, and software delivery effectiveness compared to rigid hierarchical structures.

The review also identified several socio-technical factors affecting collaboration and coordination, including communication mechanisms, psychological safety, organizational culture, human factors, distributed coordination systems, knowledge-sharing practices, and socio-technical alignment between organizational workflows and technical infrastructures.

Overall, the review demonstrates that effective software engineering organizations require strong alignment between organizational structures and socio-technical systems to sustain collaboration, coordination, software quality, and project success in modern digital environments.

The review contributes to evidence-based software engineering research by integrating organizational theory and socio-technical systems perspectives within contemporary software development environments. The findings may support software engineering managers, Agile practitioners, DevOps leaders, and organizational decision-makers in designing collaborative structures that improve coordination effectiveness and software delivery outcomes. Future research may further investigate artificial intelligence-supported coordination systems, hybrid Agile governance structures, remote collaboration ecosystems, socio-technical analytics, and organizational resilience within distributed software engineering environments.

REFERENCES

1. Abid-Baudin, M. (2022). Explaining knowledge intensive firms' performance through internal factors: Evidence from an IT consulting firm. *International Research Journal of Management Science, 7*(1). https://doi.org/10.3126/irjms.v7i1.50633
2. Al-Fraihat, D., et al. (2024). Distributed agile team coordination using intelligent learning approaches. *Heliyon, 10*, e39926. https://doi.org/10.1016/j.heliyon.2024.e39926

3. Almeida, F., & Espinheira, E. (2022). Agile governance and coordination mechanisms in large-scale software development organizations. *International Journal of Information Systems and Project Management, 10*(1), 5–24. https://doi.org/10.12821/ijispm100102
4. Buvik, M. P., & Tkalich, A. (2023). Psychological safety and team performance in agile software development teams. *Frontiers in Psychology, 13*, 1141571. https://doi.org/10.3389/fpsyg.2023.1141571
5. Dingsøyr, T., Moe, N. B., & Seim, E. A. (2024). Challenges in understanding the relationship between teamwork quality and project success in large-scale agile projects. *Empirical Software Engineering, 29*, Article 34. https://doi.org/10.1007/s10664-022-10208-4
6. Edison, H., Wang, X., & Abrahamsson, P. (2022). Large-scale agile development methodologies: A systematic literature review. *IEEE Transactions on Software Engineering, 48*(3), 830–857. https://doi.org/10.1109/TSE.2021.3069039
7. Grossman, R., et al. (2022). Should the existing science of teams be applied to fluid teams? An exploration of fluid team effectiveness within the context of healthcare simulation. *Frontiers in Artificial Intelligence, 5*, 818562. https://doi.org/10.3389/frai.2022.818562
8. Gustavsson, T. K., et al. (2022). Changes to team autonomy in large-scale software development: A multiple case study of scaled agile framework (SAFe) implementations. *Journal of Systems and Software, 191*, 111357. https://doi.org/10.1016/j.jss.2022.111357
9. Hustad, E., et al. (2022). Knowledge mobilization in agile information systems development projects: A systematic literature review. *Electronic Journal of Knowledge Management, 20*(2), 139–154. https://doi.org/10.34190/eckm.23.2.746
10. Ijaz, S., et al. (2022). Distributed agile team coordination using task allocation approaches: A systematic review. *International Journal of Advanced Computer Science and Applications, 13*(2), 589–598. https://doi.org/10.14569/IJACSA.2022.0130269
11. Iffah, N., et al. (2023). The influence of organizational, process and individual factors on software development success. *International Journal of Business and Management Studies, 3*(8), 45–58. https://doi.org/10.56734/ijbms.v3n8a6
12. Kalluri, R. (2022). A human factors study of risk management of complex agile scrum projects in large enterprises. *Informatics, 9*(1), 20. https://doi.org/10.3390/informatics9010020
13. Klymenko, O., et al. (2022). Understanding the implementation of technical measures in the process of data privacy compliance: A qualitative study. *Information Systems Frontiers, 26*, 121–139. https://doi.org/10.1007/s10799-022-00370-y
14. Kukhareva, T., et al. (2022). Coordination and communication structures in software engineering organizations. In *Proceedings of the 55th Hawaii International Conference on System Sciences* (pp. 7681–7690). https://doi.org/10.24251/HICSS.2022.880
15. Matthews, J., & Tanner, K. (2024). National culture and collaboration dynamics in agile software development teams. *Frontiers in Psychology, 14*, 1323469. https://doi.org/10.3389/fpsyg.2024.1323469
16. Moser, C., & Moura, H. (2022). Maturity models for managing people in software development teams: A systematic literature review. *Journal of Organizational and End User Computing, 34*(2), 174–196. https://doi.org/10.4018/JOEUC.304810
17. Nazir, S., et al. (2022). Adapting agile development practices for hyper-agile environments: Lessons learned from a COVID-19 emergency response research project. In *Proceedings of the ACM International Conference on Evaluation and Assessment in Software Engineering* (pp. 1–10). https://doi.org/10.1145/3544902.3546234
18. Olivares, D., et al. (2024). Team allocation optimization in agile software development using intelligent learning approaches. *Scientific Reports, 14*, Article 45415. https://doi.org/10.1038/s41598-023-45415-6
19. Pokhrel, S., & Goyal, P. (2022). Psychological workplace climate, emotional intelligence and employee well-being in the Nepali information technology industry. *Research on Humanities and Social Sciences, 12*(14), 11–19. https://doi.org/10.7176/RHSS/12-14-02
20. Tamburri, D. A., et al. (2023). Software architecture and team organization in DevOps environments: An empirical investigation. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis* (pp. 1–12). https://doi.org/10.1145/3641822.3641868

21. Trzeciak, M., et al. (2022). Enablers of open innovation in software development micro-organizations. *Journal of Open Innovation: Technology, Market, and Complexity, 8*(4), 174. https://doi.org/10.3390/joitmc8040174
22. Zhou, X., et al. (2022). A cross-company ethnographic study on software teams for DevOps and microservices: Organization, benefits, and issues. In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1–13). https://doi.org/10.1145/3510457.3513054