Alq J Sci. 2026;2(2):190-201
https://doi.org/10.69667/ajs.26204
Alqalam Journal of Science
مجلـــة القلم للعلـوم
https://alqalam.utripoli.edu.ly/index.php/AR

The Synergistic Impact of Integrated Agile and deveops on Software Delivery Performance: An Evidence-Based Synthesis of Velocity, Quality, and Organizational Scaling

Abdulgader Ab Sinus[1], Noureddon Beleid[1*], Ashrf Daw Samlghy[2]

[1]*Department of Computer Sciences, Higher Institute of Sciences and Technology, Soq Alkamis Msehel, Libya*
[2]*Department of Computer Sciences, Higher Institute of Sciences and Technology, Tripoli, Libya*
Corresponding email. *nooreb@yahoo.com*

Abstract
The domain of software project management is confronting escalating challenges in adapting to rapidly evolving market demands while ensuring the delivery of high-quality products. In this context, Agile and DevOps methodologies have emerged as pivotal frameworks for accelerating software development and enhancing product quality through fostered collaboration and the seamless integration of development and operations processes. This study provides a rigorous, evidence-based synthesis of the demonstrable impact of integrated Agile and DevOps practices on quantifiable software delivery performance metrics. Adopting a Systematic Literature Review (SLR) protocol, this paper rigorously aggregates and synthesizes empirical studies to validate the causal links between methodology adoption and high-performance outcomes across the four dimensions of the DORA (DevOps Research and Assessment) framework. The findings confirm that the synergy between Agile and DevOps leads to the simultaneous optimization of development velocity (high Deployment Frequency and reduced Lead Time for Changes) and system stability (low Change Failure Rate and reduced Mean Time to Recovery).Furthermore, the analysis reveals that successful, large-scale adoption is critically dependent on addressing cultural constraints, fostering psychological safety, integrating security via DevSecOps, and strategically choosing scaling frameworks such as SAFe or LeSS. The core conclusion is that technical rigor alone is insufficient; sustained elite performance requires a holistic transformation that balances technological advancement with profound organizational and cultural change.
Keywords. Agile; DevOps; DORA Metrics; Systematic Literature Review; Software Delivery Performance; DevSecOps.

INTRODUCTION

The contemporary landscape of software development is defined by relentless market volatility and escalating project complexity, requiring organizations to deliver high-quality products under compressed timelines. Traditional, plan-driven project management methodologies are frequently characterized by rigidity and prove inadequate in maintaining competitive relevance within such a dynamic environment. This inadequacy has catalysed a necessary paradigm shift toward iterative and adaptive approaches. Agile and DevOps have emerged as pivotal frameworks, not merely as process optimizations, but as fundamental cultural and technical philosophies designed to accelerate software development and profoundly enhance product quality. This study aims to move beyond anecdotal evidence by providing a rigorous, evidence-based synthesis of the demonstrable impact of integrated Agile and DevOps practices on quantifiable software delivery performance metrics, analysing the complex challenges associated with their implementation, and proposing advanced strategies for enterprise-scale adoption. Agile development is defined by a set of principles that promote iterative cycles, continuous customer collaboration, and a rapid, adaptive response to change (1).

Frameworks like Scrum and Kanban focus primarily on optimizing the efficiency and flexibility of the development lifecycle. Conversely, DevOps represents a cultural and technical philosophy aimed at systematically bridging the historical organizational and technical divide between development and operational processes. Its objective is to ensure rapid, stable, and reliable software deployments. Scholarly work posits that while Agile primarily optimizes the upstream "development" segment of the value stream, DevOps optimizes the downstream "delivery" and operational segment. The integrated application of these two methodologies creates a powerful, efficient, end-to-end system that encompasses the entire software delivery value stream, from concept creation to production deployment and monitoring. This integration is what facilitates the significant reduction in delivery cycle times, often reported to be over 40%, thereby enabling organizations to secure a substantial competitive advantage (Smith, 2025). While the qualitative benefits of Agile and DevOps such as improved collaboration, faster time to market, and increased customer satisfaction are widely reported [2], a significant gap persists in synthesizing the existing academic and empirical literature around a standardized, quantifiable metric system that rigorously confirms these benefits [9]. Many studies rely on internal or non-standard metrics, hindering cross-organizational comparison. This paper addresses this crucial gap by adopting the DevOps Research and Assessment (DORA) framework as the definitive set of metrics for evaluating the outcomes of Agile-DevOps adoption [10]. The DORA metrics provide a reliable, empirically backed, and standardized means of measuring both development velocity (Deployment Frequency and Lead Time for Changes) and system stability/product quality (Mean Time to Recovery and Change Failure Rate) [9]. This analysis rigorously aggregates and synthesizes empirical studies to validate the causal links between Agile-DevOps maturity, specific organizational practices, and high-performance outcomes across all four DORA dimensions. This paper is structured according to the standard academic IMRAD (Introduction, Methods, Results, and Discussion) format [15]. Section 2 establishes the theoretical foundation and reviews the relevant literature. Section 3 outlines the rigorous Systematic Literature Review (SLR) protocol adopted. Sections 4 presents the synthesized results concerning velocity and quality metric and also provide an in-depth discussion, analysing advanced implementation challenges, including organizational scaling (SAFe vs. LeSS), the integration of security (DevSecOps), the foundational role of organizational culture and leadership, synthesizes these findings and addresses potential threats to validity. Finally, Section 5 concludes the analysis and offers critical implications and future research directions.

Related work

The Evolution of Software Paradigms

The foundational shift in software engineering practice was marked by the Agile Manifesto in 2001, which prioritized "individuals and interactions over processes and tools" and "responding to change over following a plan" [5]. This marked a transition from rigid, sequential, plan-driven methodologies (e.g., Waterfall) to flexible, iterative approaches. This foundational movement led to the widespread adoption of frameworks such as Scrum, which promotes development in short, time-boxed increments (sprints) using cross-functional teams, and Kanban, which focuses on visualizing workflow and managing flow control [37]. The literature confirms the efficacy of these frameworks in enhancing stakeholder satisfaction and managing rapidly changing requirements. The core of Agile methodology is predicated on continuous customer feedback and iterative development, significantly enhancing the capacity to accommodate evolving project requirements [7].

The Emergence and Principles of DevOps

The advent of Agile, while optimizing development speed, often amplified the inherent organizational friction between the speed-focused development teams and the stability-focused operations teams. This friction frequently resulted in deployment bottlenecks and instability, leading to the emergence of DevOps. DevOps is defined as a philosophy focused on extensive automation, Continuous Integration (CI), Continuous Delivery (CD), and proactive monitoring to dismantle organizational silos and expedite service delivery. Key works in the field articulate the core tenets of DevOps as a culture of shared responsibility, amplifying feedback loops, and pursuing continuous experimentation and learning. The successful implementation of DevOps practices, particularly continuous deployment through automated CI/CD pipelines, directly addresses historical bottlenecks, empirically leading to higher deployment frequencies and significantly shorter lead times for code changes [37].

Measuring Modern Software Delivery Performance

A critical element of the DevOps philosophy is Measurement (the 'M' in the CALMS mnemonic), demanding that performance be tracked objectively. The DORA framework, developed by the DevOps Research and Assessment team, provides a reliable and standard set of metrics for evaluating process performance and maturity [10]. These four core metrics provide a holistic view by measuring both the speed (velocity) and the stability (quality) of the software delivery pipeline, overcoming the limitations of tracking only development-centric metrics [20].

DORA Velocity Metrics

The velocity dimension is captured by two metrics:

i. **Deployment Frequency (DF):** This measures how often code is successfully deployed to a production environment. High DF is considered the most relevant measurement of DevOps success, showcasing an organization's efficiency and responsiveness to user needs [18].
ii. **Lead Time for Changes (LTTC):** This measures the duration from the time a change is committed to the version control system until it is successfully running in production. LTTC is a key indicator of lean efficiency and the end-to-end velocity of the combined Agile-DevOps system [9].

DORA Quality and Stability Metrics

The stability dimension, essential for ensuring that speed does not compromise reliability, is captured by:

i. **Change Failure Rate (CFR):** This metric tracks the percentage of changes or deployments that result in degraded service, requiring remediation or a fix. It directly measures the quality and correctness of the pipeline and the deployed code [10].
ii. **Mean Time to Recovery (MTTR):** This measures the average time it takes for an organization to restore service after a production failure or system outage. Low MTTR is a critical measure of operational resilience and end-customer experience [10].

The fundamental principle derived from DORA research is that high-performing teams must excel across all four metrics simultaneously, confirming that successful Agile-DevOps implementation requires achieving speed *and* stability, not one at the expense of the other [10].

Existing Literature on Challenges

The literature confirms that the transition to integrated Agile-DevOps is fraught with organizational and technical challenges. Cultural barriers often represent the most significant hurdle, particularly the historical divide and friction between development and operations teams, compounded by general internal resistance to change [28]. Technical difficulties include the complexity of integrating diverse toolchains, the necessity of modernizing legacy infrastructure, and, critically, maintaining adequate governance and security compliance without impeding the desired flexibility and speed. Importantly, existing research emphasizes that technical practices alone are insufficient; sustained success mandates a supportive organizational culture that champions psychological safety, organizational learning, and risk-taking [28].

Methods

This study adopts a systematic review methodology guided by the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) framework to ensure methodological rigor, transparency in study selection and reporting, and the replicability of the review process. A Systematic Literature Review (SLR) is selected as the appropriate methodology [24]. An SLR systematically aggregates, evaluates, and synthesizes existing empirical evidence from high-impact peer-reviewed literature relating Agile-DevOps practices to quantifiable performance outcomes [25]. This approach ensures transparency, replicability, and a structured foundation for drawing empirically grounded conclusions regarding the causal relationships between methodology adoption and improved performance metrics, meeting the standards of academic publication [15].

Research Questions

The SLR is guided by the following focused research questions:

RQ1: What empirical evidence exists regarding the correlation between Agile-DevOps maturity and software delivery velocity metrics (Deployment Frequency and Lead Time for Changes)?

RQ2: What empirical evidence exists regarding the correlation between Agile-DevOps practices and product quality/stability metrics (Mean Time to Recovery and Change Failure Rate)?

RQ3: What organizational, cultural, and technical challenges impede the realization of high Agile-DevOps performance, and how do recognize scaling frameworks (SAFe, LeSS) and integrated security practices (DevSecOps) mitigate these impediments?

SLR Protocol Definition
The formal protocol defines the systematic steps used to identify, screen, and synthesize the relevant body of knowledge [25].

Search Strategy
The research sought articles from recognized databases specialising in software engineering and information systems, including IEEE Xplore, the ACM Digital Library, SpringerLink, and ScienceDirect, covering peer-reviewed publications between 2017 and 2024, to ensure temporal relevance [35].
Primary search strings utilized combinations of keywords relating the methodology to the quantifiable outcome: ("DevOps" OR "Agile") AND ("DORA metrics" OR "Deployment Frequency" OR "Lead Time for Changes" OR "MTTR" OR "Change Failure Rate") AND ("empirical" OR "correlation" OR "case study" OR "systematic literature review").

Eligibility Criteria
The criteria for inclusion mandated that studies must be: [1] peer-reviewed journal articles, conference papers, or comprehensive systematic literature reviews; [2] empirical in nature (case studies, surveys, experiments, or quantitative analyses); and [3] explicitly link the adoption or maturity of Agile and/or DevOps practices to quantifiable performance metrics, ideally aligning with the DORA framework or directly comparable velocity/quality measures. Exclusion criteria focused on papers lacking empirical data, purely conceptual or opinion-based articles, and studies published before 2017.

Screening and Selection process
The database search identified 42 records. After the removal of five duplicate records, 29 records remained for title and abstract screening. Of these, five records were excluded for not meeting the scope of the review. The full texts of 29 articles were subsequently assessed for eligibility based on the predefined inclusion and exclusion criteria. As a result, 25 studies met the eligibility criteria and were included in the final review. The study selection process is presented in Figure 1 using the PRISMA flow diagram, which illustrates the number of records identified, screened, excluded, and included at each stage.

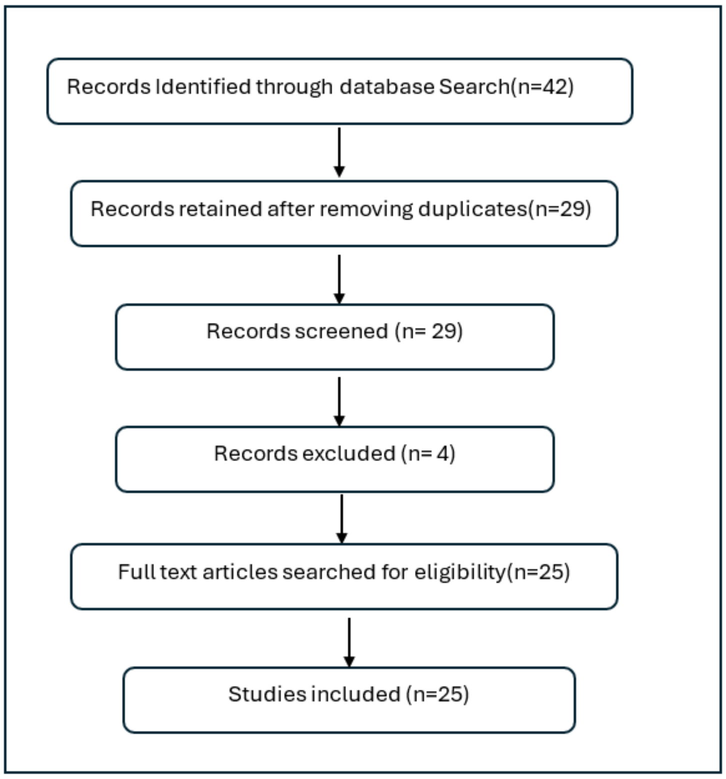
Records Identified through database Search(n=42)
Records retained after removing duplicates(n=29)
Records screened (n= 29)
Records excluded (n= 4)
Full text articles searched for eligibility(n=25)
Studies included (n=25)

Figure 1. PRISMA flow diagram

Data Extraction and Synthesis

A structured data extraction plan was utilized to collect key metadata (author, year, primary study methodology) and specific outcome data, including correlation coefficients (ρ), reported average metric improvements, and qualitative descriptions of benefits and challenges [25]. The synthesis employed a conceptual approach, grouping and mapping the reported quantitative findings to the DORA framework dimensions (velocity vs. quality) [26]. This focused approach ensures that the "Results" section is grounded in objective data, moving beyond the anecdotal claims found in general industry reports.

Results and Discussion

Impact on Development Velocity (DF and LTTC)

The integration of Agile and DevOps has a validated, measurable impact on software delivery velocity, evidenced primarily through substantial improvements in Deployment Frequency (DF) and reduction in Lead Time for Changes (LTTC).

Deployment Frequency (DF)

Deployment Frequency DF is recognized as a principal metric for gauging organizational responsiveness and the successful implementation of Continuous Integration and Continuous Delivery (CI/CD) practices [18]. The literature confirms a strong, positive relationship between organizational maturity in integrated Agile-DevOps and high DF. For instance, mixed-methods studies, combining systematic reviews with multiple embedded case studies, have demonstrated a significant correlation between a measured maturity score based on Lean-Agile DevOps frameworks and Deployment Frequency, reporting a correlation coefficient of $\varrho = 0.67$, $p < 0.01$) and a 31% reduction in MTTR [35]. This robust quantitative evidence validates that improved process capability directly translates into the ability to release software quickly and often. The role of Agile in this velocity gain is through minimizing the "batch size" of work via iterative sprints and continuous integration, ensuring that deployable code is consistently produced.[1] This continuous flow is then leveraged by the DevOps practice of CI/CD automation, which eliminates manual processes and bottlenecks in the pipeline, enabling organizations to achieve elite performance benchmarks of multiple deployments per day or even on-demand deployment [10]. The measured increase in DF is an indicator of efficiency and the speed at which end-users receive value [18].

Lead Time for Changes (LTTC)

L Lead Time for Changes TTC, measuring the time from code commitment to production release, is a definitive metric for the efficiency of the combined Agile development and DevOps delivery system [9]. High-performing teams consistently achieve LTTCs under one hour, indicating that the delivery process is highly automated and friction-free [10]. Empirical data confirms the enormous potential for efficiency gains, with organizations applying Lean/DevOps principles observing reductions in the overall Time to Value by up to 55% [21]. Analysis of the delivery pipeline reveals that process friction, often stemming from human activities rather than technical limitations, presents the greatest drag on LTTC. Specifically, the Pull Request (PR) review process is frequently identified as a significant source of "hidden delay," potentially accounting for approximately 30% of the total Cycle Time [21]. This finding is critical because it highlights that technical automation (DevOps) alone is insufficient; teams must also apply Lean principles to optimize the human-centric aspects of the Agile workflow, such as reducing PR size and improving code review velocity (K, 2025). The definitive metric for the combined efficiency of the Agile development stream and the DevOps delivery pipeline is LTTC improvement. If high deployment frequency is achieved, yet LTTC remains long, it signals organizational or human process drag that must be addressed through optimization efforts rooted in Lean thinking, rather than merely adding more automation. Table 1 summarizes the DORA framework and the core practices enabling high performance across these metrics.

Table 1: The DORA Metrics Framework and Correlation to Agile-DevOps Practices

Performance Dimension	Key Metric	Elite Performance Benchmark	Core Agile-DevOps Enabler
Velocity	Deployment Frequency (DF)	Multiple deploys per day	Continuous Integration/Continuous Delivery (CI/CD) Automation [10]
Velocity	Lead Time for Changes (LTTC)	Less than one hour	Lean flow principles and small batch sizes [9]
Quality	Mean Time to Recovery (MTTR)	Less than one hour	Automated alerting and rapid rollback capabilities [10]
Quality	Change Failure Rate (CFR)	0–15%	Shift-Left testing, quality integration, and test automation [20]

Impact on Product Quality and Stability (MTTR and CFR)
The second critical finding is that the synergy between Agile and DevOps does not sacrifice stability for velocity; instead, it optimizes both simultaneously. This is demonstrated by significant improvements in stability metrics (CFR and MTTR).

Change Failure Rate (CFR) and Defect Density
CFR, the rate at which deployments fail and require immediate remediation, is the direct indicator of product quality within a continuous delivery environment [20]. Elite performing organizations maintain a low CFR, typically in the 0–15% range [10]. This low rate is fundamentally achieved by integrating quality activities early in the Software Development Lifecycle (SDLC). Agile facilitates this through continuous testing within each short sprint, which prevents the systemic accumulation of defects over time.
DevOps technical practices amplify this preventative approach. Empirical studies confirm that implementing technical practices such as unit testing, test-driven development (TDD), and static analysis significantly help to inject fewer defects and decrease remediation costs [14]. This preventative strategy is commonly referred to as "Shift-Left Quality," where security checks, performance testing, and code scanning are automated and performed directly within the CI/CD pipeline, catching defects when the cost of remediation is lowest.[1] The quantitative success of this integration is observed in data indicating that organizations implementing both methodologies report up to a 35% reduction in error rates.

Mean Time to Recovery (MTTR)
MTTR measures the operational resilience of a system by tracking the average time required to restore service after a system failure. Elite organizations target MTTRs of less than one hour, ensuring rapid business continuity [10]. This resilience is enabled by DevOps practices, including automated alerting, comprehensive post-deployment monitoring, and the extensive use of infrastructure-as-code (IaC) templates, which allow for rapid, standardized rollback capabilities.Case study evidence demonstrates that prioritizing improvements in stability metrics yields substantial results. One case study noted that by focusing on improving code coverage (increasing from 65% to 82%) and reducing code smells (by 40%), the organization achieved elite DORA status. These improvements resulted in a 37% increase in quality, marked by significantly faster recovery times and fewer deployment issues, moving their MTTR and CFR from "high" to "elite" performance bands [13]. The evidence underscores that the success of the combined Agile-DevOps model is measured by the *simultaneous* optimization of all four DORA metrics. The analysis demonstrates that any attempt to maximize velocity (DF) without commensurate investment in quality and stability controls (CFR, MTTR) ultimately proves counterproductive, leading to organizational instability and negating the competitive advantage derived from speed. High performance is achieved when iterative, quality-focused development (Agile) is matched by rapid, resilient, automated operations (DevOps).

Advanced Implementation: Navigating Organizational Scaling
As organizations mature in their adoption of Agile and DevOps, they inevitably confront the challenge of scaling these practices across large, often globally distributed, enterprises. Large organizations encounter unique challenges,

including internal resistance stemming from entrenched bureaucracy, the mandatory restructuring of teams, and the difficulty of defining a standardized requirements framework across hundreds or thousands of individuals [1]. These complexities necessitate the adoption of formal scaling frameworks to maintain coherence and alignment across numerous teams.

Comparative Analysis of Scaling Frameworks (SAFe vs. LeSS)

The choice of a scaling framework represents a fundamental strategic decision regarding organizational architecture: complexity for control versus simplicity for agility. The two leading frameworks are the Scaled Agile Framework (SAFe) and Large-Scale Scrum (LeSS).

i. The Scaled Agile Framework (SAFe)

SAFe is a prescriptive, highly structured framework suitable for very large companies that prioritize aligning strategic planning with execution across multiple, interdependent teams. SAFe addresses complexity by introducing defined layers (Portfolio, Program, Team) and formalizing roles, most notably the Agile Release Train (ART) [36]. The advantages of SAFe include significant productivity boosts (studies show up to 30–35% improvement) and better strategic alignment, as the framework ensures that leadership goals connect directly with team execution [16]. SAFe's structure streamlines workflows, enabling quicker responses to market demands [1]. However, implementing SAFe requires substantial organizational restructuring, a high financial commitment, a longer transition period, and addressing significant cultural resistance to change due to the imposition of new roles and processes [36],[41],[42].

ii. Large-Scale Scrum (LeSS)

LeSS, by contrast, focuses on applying the foundational principles of Scrum across multiple teams working on a single product, striving to simplify organizational structure rather than adding complexity [29]. LeSS minimizes organizational overhead, maintaining a single Product Backlog and a single Product Owner for all teams [31]. LeSS emphasizes the use of self-contained Feature Teams, which are fully capable of delivering a customer-centric backlog item from start to finish [38]. Benefits of LeSS include reduced organizational complexity, enhanced cross-team collaboration, and a simpler transition for organizations already mature in basic Scrum practices [36]. The framework radically changes managerial focus from work assignment to capability-building and team enablement, eliminating middle and first-line management roles where possible [38]. LeSS views dependencies as undesirable signs of organizational weakness that must be minimized through self-coordination and close collaboration [38]. The choice between SAFe and LeSS illustrates a key strategic trade-off: organizations must determine if they require a structured, top-down governance model (SAFe) to manage complexity or if they are willing and able to undergo the radical managerial and cultural simplification necessary for LeSS to succeed [29]. The solution to scaling complexity is itself complex, forcing alignment between the chosen framework and the necessary organizational culture and structure. Table 2 details the comparative trade-offs between the two leading frameworks.

Table 2: Strategic Comparison of Leading Agile Scaling Frameworks

Framework Criteria	SAFe (Scaled Agile Framework)	LeSS (Large-Scale Scrum)
Organizational Philosophy	Alignment, Governance, and Predictability (Top-down structure)	Simplification and Decentralization (Bottom-up Scrum focus) [29]
Key Structural Element	Agile Release Trains (ARTs) and Defined Portfolio Layer	Single Product Backlog, Feature Teams [31]
Organizational Cost/Complexity	High significant restructuring, specialized roles, longer transition [36]	Low extension of Scrum, minimized managerial overhead [38]
Dependency Management	Managed primarily through formal coordination and planning (e.g., PI Planning)	Viewed as a weakness; teams are responsible for self-coordination and minimizing dependencies [38]

Advanced Implementation: Integrating Security via DevSecOps

Security concerns frequently present a major functional hurdle to achieving high-velocity DevOps.[1] The necessity for rigorous security checks often risks becoming a manual bottleneck that undermines the velocity gains of CI/CD. The integrated solution is DevSecOps, a practice that transforms security from a traditional organizational gatekeeper function into a continuous, parallel, and shared responsibility throughout the SDLC [30].

The "Shift-Left" Paradigm
DevSecOps fundamentally relies on the concept of "Shifting Left" moving security validation, testing, and compliance checks to the earliest possible stages of development, such as the code commit and build time [34],[40]. This strategy ensures that security flaws and misconfigurations are identified when the cost of remediation is lowest [32]. By automating security integration, organizations ensure that the rapid deployment frequency enabled by Agile-DevOps does not compromise system integrity, thereby preserving the gains in system stability (low CFR and MTTR). If security checks remain manual and late in the cycle, they impose delay (long LTTC) or lead to massive post-deployment failures (high CFR/MTTR). DevSecOps is the necessary bridging mechanism between high velocity and continuous quality.

Key DevSecOps Practices in the CI/CD Pipeline
Specific practices and tools are integrated into the automated CI/CD pipeline to ensure security is continuous and frictionless [11]:

- **Automated Security Scanning:** Static Application Security Testing (SAST) is integrated during the code review process to identify vulnerabilities in the source code before compilation [34]. Dynamic Application Security Testing (DAST) is run automatically in staging or pre-production environments to detect runtime vulnerabilities.
- **Dependency and Artifact Management:** Automated container image scanning and Software Bill of Materials (SBOM) generation are utilized to track and inventory all dependencies and expose known vulnerabilities [34]. Preventing vulnerable components from reaching production is crucial for maintaining a low Change Failure Rate.
- **Policy-as-Code and Configuration Scanning:** Infrastructure-as-Code (IaC) tools are scanned to ensure compliant configurations before deployment. This preventative measure prevents security misconfigurations, which can frequently be the cause of outages or breaches, thus directly contributing to a lower Mean Time to Recovery [11]. Table 3 provides an overview of how DevSecOps practices are strategically placed within the pipeline to support DORA metrics.

Table 3: DevSecOps "Shift-Left" Integration Practices Across the CI/CD Pipeline

SDLC Stage	DevSecOps Practice	Impact on Performance Metrics	Value Proposition
Code/Build	Static Analysis (SAST) and Dependency Scanning	Lowers Defect Density; Reduces CFR	Finds security bugs and known vulnerabilities before compilation [34].
Configuration	IaC Scanning / Policy-as-Code	Ensures compliance and stability; Reduces MTTR	Prevents deployment of known misconfigurations that cause outages or breaches [11].
Test/Stage	Dynamic Analysis (DAST) and Fuzz Testing	Lowers CFR	Identifies runtime vulnerabilities in a safe, automated environment.
Deploy/Release	Container Image Scanning and Artifact Signing	Reduces CFR; Improves deployment reliability	Prevents deployment of known vulnerable components to production [34].

The Crucial Role of Organizational Culture and Leadership
The most sophisticated technology and the most rigorous process protocols will fail without the requisite supportive organizational culture. Cultural factors are recognized as the primary constraint on achieving peak Agile-DevOps performance.

Cultural Barriers to Agile-DevOps Success
The existing literature highlights those organizational cultures rooted in traditional hierarchical structures and a

pervasive resistance to change pose significant obstacles to transformation [22]. Challenges include overcoming deeply ingrained departmental silos, navigating resistance from employees unfamiliar with continuous feedback loops, and a lack of comprehensive knowledge regarding Agile methodologies across the organization.[1] Successfully overcoming this resistance necessitates strong executive sponsorship, continuous education, and fostering cross-departmental collaboration to dismantle the cultural and technical walls [28].

Psychological Safety as a Predictor of Performance

A collaborative and proactive culture is not merely beneficial; it is a prerequisite for success. Psychological safety defined as a shared belief that the team is safe for interpersonal risk-taking, where team members feel accepted for their true selves and safe to ask questions, experiment, and make mistakes is identified as a critical success factor.

DORA research provides empirical evidence confirming that a high-trust, generative culture is strongly predictive of high software delivery performance and organizational productivity [6]. This organizational mindset, which prioritizes curiosity and learning over blame and shame, acts as an enabling mechanism for continuous improvement [23], A lack of psychological safety manifests directly in poor DORA metrics. When engineers fear being blamed for failures, they may delay necessary changes (low Deployment Frequency/throughput) or communicate inadequately during outages, leading to prolonged Mean Time to Recovery [23]. Therefore, if technical success is dependent on continuous learning and experimentation, psychological safety is the foundational mechanism that allows that learning process to occur and directly impacts system resilience and velocity metrics.

The Evolution of Leadership

The cultural shift required to scale Agile and DevOps demands a corresponding evolution in leadership style. The transition requires adopting a Lean-Agile mindset and emphasizing servant leadership [4]. Leaders must fundamentally change their focus from managing individuals and assigning work to capability-building and enabling teams [38]. This involves "scaling intent" where leaders set the strategic objectives and vision, but trust and empower their teams to determine the best means of execution [17]. This approach supports the fundamental principles of accountability and autonomy, which are key to successful scaling and sustained high performance [4]. Leaders must be prepared to challenge the status quo, embrace the idea of "failing fast," and tailor the Agile approach to the unique needs of different teams [10].

Threats to Validity

Synthesis of Findings

The synthesis of existing literature unequivocally validates the central proposition: the integrated, synergistic application of Agile and DevOps practices profoundly enhances software delivery performance. This enhancement is not limited to perceived qualitative gains but is measurable through the simultaneous optimization of the four DORA metrics. High velocity (increased DF, reduced LTTC) is achieved through Lean principles that minimize batch size (Agile) and technical automation of the deployment pipeline (DevOps). This speed is maintained without sacrificing quality because security and testing are integrated early (DevSecOps Shift-Left), leading to minimal defects (low CFR) and high resilience (low MTTR). However, the analysis demonstrates that optimizing process and technology is insufficient. The cultural framework specifically, the existence of psychological safety and supportive, enabling leadership serves as the primary constraint on achieving elite performance. If cultural barriers persist, teams will inevitably generate bottlenecks (e.g., delayed reviews, poor communication during incidents) that undermine the full potential of technical automation and process rigor, leading to suboptimal LTTC and MTTR outcomes.

Threats to Validity

As an SLR, the findings are constrained by the quality and methodology of the primary studies synthesized. One primary threat to internal validity is the difficulty in isolating the distinct impacts of Agile versus DevOps, as they are often implemented concurrently, making it challenging for researchers to attribute performance changes definitively to one paradigm over the other [26]. Furthermore, many empirical studies, particularly surveys and case studies, rely on self-reported data by participants regarding their practices or maturity levels, which introduces the potential for positive response bias [35]. Finally, the rapidly evolving nature of the field, particularly concerning DevSecOps, means that best practices are constantly changing, posing a temporal threat to the longevity of the findings regarding specific technical toolchains and practices [30].

Conclusion

This analysis confirms that Agile and DevOps represent the most transformative, evidence-backed trends in modern software project management. The rigorous application of both methodologies, validated through the DORA performance framework, proves that organizations can achieve exceptional speed (high DF, low LTTC) and stability (low CFR, low MTTR) concurrently. The core contribution of this synthesis is the confirmation that this synergy is attained only through a holistic approach that balances technical investment with profound organizational and cultural transformation. Addressing cultural constraints, implementing appropriate scaling models (SAFe/LeSS), and integrating security as a core, automated capability (DevSecOps) are non-negotiable requirements for achieving sustainable, high-level software delivery performance. The findings offer several critical implications for organizations undertaking or struggling with Agile-DevOps transformations: Firstly, organizations must adopt a Lean focus on reducing Lead Time for Changes as the definitive measure of flow efficiency. This requires addressing friction in human processes, such as code review bottlenecks, by encouraging small, frequent changes and investing in the automation of the entire value stream [21]. Secondly, deployment Frequency should never be tracked in isolation. Practitioners must use the four DORA metrics as a balanced scorecard. Over-optimization of velocity at the expense of stability leads to increased Change Failure Rate and Mean Time to Recovery, negating the business value of speed (K, 2025). Moreover, senior leadership must proactively cultivate a generative, high-trust culture. This means eliminating blame following incidents and fostering an environment where teams feel safe to experiment, learn from failure, and rapidly communicate during critical outages, directly impacting MTTR [23]. Last but not least, security must be automated and shifted left into the CI/CD pipeline using DevSecOps tools (SAST, IaC scanning). Treating security as an automated preventative control, rather than a late-stage gatekeeper, is essential to maintaining high velocity and low CFR [34].

Future Research Directions

Based on the identified gaps and limitations in the current body of literature, several directions for future research are warranted and there as follows:

i. Large-Scale Quantitative Validation: Future work should prioritize quantitative studies utilizing large, cross-sectoral datasets to further identify the specific advantages of the combined Agile-DevOps adoption, controlling for organizational size and industry sector [2]. Rigorous studies are required to explore the correlation between specific dimensions of Agile and DevOps maturity and the resultant DORA performance gains [35].
ii. Automated Maturity Modeling and Telemetry: Research should focus on developing objective maturity models that move beyond self-reporting. This includes creating automated telemetry systems capable of integrating DORA metrics, code quality indicators, and objective measurements of cultural health (e.g., communication patterns, blameless post-mortem data) to provide real-time, actionable insights into transformation effectiveness [1].
iii. Integration of AI and Machine Learning: Further applied research is needed to explore the integration of Artificial Intelligence and Machine Learning into the Agile-DevOps pipeline for predictive analytics, intelligent resource allocation within sprints, and optimizing release risk assessment, thereby creating more intelligent and efficient systems.
iv. Alignment with IT Service Management (ITSM): Research should investigate the integration and strategic alignment of Agile and DevOps practices with traditional enterprise IT Service Management frameworks, such as ITIL. This effort would aim to define practical frameworks that ensure enterprise agility and Lean principles are consistently applied across the full IT service lifecycle, addressing challenges and outlining future research directions regarding this critical institutional alignment [39].

Conflict of interest. Nil

Received*: 02-12-2025 -* ***Accepted****: 30-01-2025 -* ***Published****: 08-02-2026*

References

1. AgileFever. Safe framework: pros and cons guide [Internet]. 2025 Mar 6 [cited 2026 Feb 8]. Available from: https://www.agilefever.com/the-ultimate-guide-to-pros-and-cons-safe-framework/
2. Almeida F, Simões J, Lopes S. Exploring the benefits of combining DevOps and Agile. Future Internet. 2022;14(2):63. https://doi.org/10.3390/fi14020063
3. Atlassian. Dora metrics: how to measure open DevOps success [Internet]. 2025 [cited 2026 Feb 8]. Available from: https://www.atlassian.com/devops/frameworks/dora-metrics
4. Appelbaum B. Scaling agile: how to overcome 3 common challenges [Internet]. Planview; 2022 [cited 2026 Feb 8]. Available from: https://www.planview.com/resources/guide/what-is-agile-program-management/scaling-agile-common-challen/
5. Beck K, Beedle M, van Bennekum A, Cockburn A, Cunningham W, Fowler M, et al. Manifesto for agile software development [Internet]. 2001 [cited 2026 Feb 8]. Available from: https://agilemanifesto.org/
6. DORA.DEV. Capabilities: generative organizational culture [Internet]. 2025 [cited 2026 Feb 8]. Available from: https://dora.dev/capabilities/generative-organizational-culture/
7. Dingsøyr T, Moe NB, Seim EA. Coordinating knowledge work in multi-team programs: a systematic review of agile software development. J Syst Softw. 2021;174:110881.
8. Faustino JP, Adriano D, Amaro R, Pereira RD, Silva MM. DevOps benefits: a systematic literature review. Softw Pract Exp. 2022;52:1905-26.
9. Gebrewold Y, Wirell J. Challenges with measuring software delivery performance metrics: a case study at a software organisation [dissertation on the Internet]. 2023 [cited 2026 Feb 8]. Available from: https://urn.kb.se/resolve?urn=urn:nbn:se:lnu:diva-121933
10. Bruneaux T. What are Dora metrics? complete guide to measuring DevOps performance [Internet]. DX; 2025 [cited 2026 Feb 8]. Available from: https://getdx.com/blog/dora-metrics
11. US Department of Defense. DevSecOps fundamentals guidebook: DevSecOps tools & activities [Internet]. 2021 Dec 14 [cited 2026 Feb 8]. Available from: https://dodcio.defense.gov/Portals/0/Documents/Library/DevSecOpsTools-ActivitiesGuidebook.pdf
12. D K, Holdowsky J. When scaling agile, engaged self-aware leadership matters. Deloitte Insights [Internet]. 2025 [cited 2026 Feb 8]. Available from: https://www.deloitte.com/us/en/insights/topics/business-strategy-growth/scaling-agile-leadership-principles-for-agile-success.html (deloitte.com in Bing)
13. Kulkarni PA. [Case study] how Socly's team used Dora metrics to achieve elite engineering performance [Internet]. DevDynamics Blog; 2024 Nov 12 [cited 2026 Feb 8]. Available from: https://devdynamics.ai/blog/socly-elite-dora-status/
14. Kavuri S. Shift-left and shift-right testing approaches: a practical roadmap for continuous quality in agile and DevOps. J Inf Syst Eng Manag. 2024;9(4). https://doi.org/10.52783/jisem.v9i4.127
15. ICCK. Instructions for authors – ICCK journal of software engineering [Internet]. 2025 Jan 2 [cited 2026 Feb 8]. Available from: https://www.icck.org/jse/instructions-for-authors
16. HelloSM. Top scaled agile framework (SAFE) challenges & benefits [Internet]. 2025 [cited 2026 Feb 8]. Available from: https://hellosm.in/mastering-safe-how-to-overcome-scaled-agile-challenges/
17. Lustig M. Scaled intent leadership: how to scale agile by scaling meaning and purpose [Internet]. Scrum Alliance; 2023 Mar 24 [cited 2026 Feb 8]. Available from: https://resources.scrumalliance.org/Article/scaled-intent-leadership
18. Waydev. Deployment frequency [Internet]. n.d. [cited 2026 Feb 8]. Available from: https://waydev.co/wp-content/uploads/2022/07/Deployment-Frequency_compressed.pdf
19. e-Informatica. E-informatica software engineering journal. E-Informatica Softw Eng J. 2025. https://doi.org/10.37190/e-inf
20. Pearson BL. What are Dora metrics and how to unlock elite engineering performance [Internet]. LINEARB Blog; 2025 Apr 8 [cited 2026 Feb 8]. Available from: https://linearb.io/blog/dora-metrics
21. Plandek. Applying the Dora metrics to drive DevOps success [Internet]. 2023 [cited 2026 Feb 8]. Available from: https://plandek.com/wp-content/uploads/2023/06/Plandek-DORA-Metrics-Whitepaper.pdf
22. Ndou V, Ingrosso A, Di Girolamo A. Framework for agile transformation: guiding organizations through cultural, structural, and competency shifts in project management. Adm Sci. 2024;14(11):301. https://doi.org/10.3390/admsci14110301
23. Software.com. . Psychological safety at work: does trust drive innovation? DevOps culture [Internet]. 2024 [cited 2026 Feb 8]. Available from: https://www.software.com/devops-guides/psychological-safety
24. Rütz M, Wedel F. DevOps: a systematic literature review. In: Proceedings of the Twenty-Seventh European Conference on Information Systems (ECIS2019); 2019 Aug; Stockholm-Uppsala, Sweden. Vol. 1.
25. Pluto Labs. How to conduct a systematic literature review in 10 steps [Internet]. Scinapse Blog; 2025 May 13 [cited 2026 Feb 8]. Available from: https://insights.pluto.im/a-step-by-step-guide-to-conducting-a-systematic-literature-review/
26. Faustino J, Adriano D, Amaro R, Pereira R, da Silva MM. DevOps benefits: a systematic literature review. Softw Pract Exp. 2022;52(9):1905-26. https://doi.org/10.1002/spe.3096
27. Fitzgerald B, Stol KJ. Continuous software engineering: a roadmap and agenda. J Syst Softw. 2017;123:176-89.

Received*: 02-12-2025 -* ***Accepted****: 30-01-2025 -* ***Published****: 08-02-2026*

Alqalam Journal of Science
مجلـــة القلم للعلوم
https://alqalam.utripoli.edu.ly/index.php/AR

28. Miller E. Understanding 5 key challenges in DevOps adoption [Internet]. Invensis Learning Blog; 2025 May 21 [cited 2026 Feb 8]. Available from: https://www.invensislearning.com/blog/devops-adoption-challenges-and-solution/
29. Monday.com. Intro to large scale scrum (LeSS) [Internet]. 2022 Jun 13 [cited 2026 Feb 8]. Available from: https://monday.com/blog/rnd/large-scale-scrum/
30. Myrbakken H, Colomo-Palacios R. DevSecOps: a multivocal literature review. Commun Comput Inf Sci. 2017;17-29. https://doi.org/10.1007/978-3-319-67383-7_2
31. Peterka P. Large scale scrum (LeSS): the ultimate guide to scaling agile [Internet]. SixSigma.us; 2024 Dec 12 [cited 2026 Feb 8]. Available from: https://www.6sigma.us/scrum/large-scale-scrum-less/
32. Pitts CCP, Catelli C, Pesek R, Pitts D, Hands J, Bednarz J, et al. Shifting left at enterprise scale: how we manage Cloudflare with infrastructure as code [Internet]. Cloudflare Blog; 2025 Dec 10 [cited 2026 Feb 8]. Available from: https://blog.cloudflare.com/shift-left-enterprise-scale/
33. Poornima K. Agile DevOps metrics every online tech team should track [Internet]. Testsigma Blog; 2025 Nov 13 [cited 2026 Feb 8]. Available from: https://testsigma.com/blog/agile-devops-metrics-to-track/
34. Queen C. What is shift left? security, testing & more explained [Internet]. CrowdStrike; 2024 Nov 26 [cited 2026 Feb 8]. Available from: https://www.crowdstrike.com/en-us/cybersecurity-101/cloud-security/shift-left-security/
35. Sethupathy UK. Navigating continuous improvement: an in-depth analysis of lean-agile and DevOps maturity models. J Softw Eng Appl. 2025;18(9):317-35. https://doi.org/10