Journix: Journal of Informatics and Computing
Vol. 1 No. 1, 2025 : 21-29
ISSN : 3090-6784 | DOI : 10.63866/journix.v1i1.4

Evolving DevOps Practices in Modern Software Engineering: Trends, Challenges, and Impacts on Quality and Delivery Performance

Zumhur Alamin [1*], Dahlan [1], Khaeruddin [2], Sahrul Ramadhan [1]

[1] Universitas Muhammadiyah Bima, Bima, Indonesia
[2] Universitas Insan Budi Utomo, Malang, Indonesia

Email: zumhur@umbima.ac.id
(* : corresponding author)

ABSTRACT − The adoption of DevOps has significantly reshaped modern software engineering by tightly integrating development and operations activities to achieve faster, more reliable, and scalable software delivery. Despite widespread industry adoption, empirical research on the quantifiable impact of DevOps practices remains fragmented. This study aims to investigate how contemporary DevOps practices influence software quality, delivery speed, and team productivity in real-world environments. A mixed-method approach was employed, combining a multi-case study across four mid-to-large enterprises with a controlled simulation of key DevOps performance metrics. Data were collected from interviews, DevOps telemetry tools (e.g., Jenkins, GitLab CI/CD, Docker, and Kubernetes), and automated system logs. The findings indicate that mature DevOps adoption is associated with a 45% increase in deployment frequency, a 38% reduction in lead time for changes, and a 32% decrease in change failure rates. Nonetheless, organizations face persistent challenges in integrating security (DevSecOps), managing technical debt, and fostering cultural alignment across siloed teams. The results confirm the strategic importance of Infrastructure as Code (IaC), continuous monitoring, and automated testing in optimizing DevOps outcomes. For practitioners, this study offers a practical roadmap for scalable and sustainable DevOps transformation. For researchers, it identifies critical gaps in socio-technical integration and intelligent automation that warrant further investigation. This research contributes to bridging empirical evidence and theoretical frameworks in the evolving field of DevOps in software engineering.

KEYWORDS: DevOps, Continuous Integration, Software Delivery, Infrastructure as Code, Team Productivity

Praktik DevOps dalam Rekayasa Perangkat Lunak Modern: Tren, Tantangan, dan Dampak terhadap Kualitas dan Kinerja Pengiriman

ABSTRAK − Adopsi DevOps telah secara signifikan membentuk kembali rekayasa perangkat lunak modern dengan mengintegrasikan aktivitas pengembangan dan operasi secara erat untuk mencapai pengiriman perangkat lunak yang lebih cepat, lebih andal, dan dapat diskalakan. Meskipun adopsi industri telah meluas, penelitian empiris tentang dampak terukur dari praktik DevOps masih terfragmentasi. Penelitian ini bertujuan untuk menyelidiki bagaimana praktik DevOps kontemporer memengaruhi kualitas perangkat lunak, kecepatan pengiriman, dan produktivitas tim di lingkungan dunia nyata. Pendekatan metode campuran digunakan, menggabungkan studi multi-kasus di empat perusahaan menengah ke atas dengan simulasi terkontrol dari metrik kinerja DevOps utama. Data dikumpulkan dari wawancara, alat telemetri DevOps (misalnya, Jenkins, GitLab CI/CD, Docker, dan

Kubernetes), dan log sistem otomatis. Temuan menunjukkan bahwa adopsi DevOps yang matang dikaitkan dengan peningkatan frekuensi penerapan sebesar 45%, pengurangan waktu tunggu perubahan sebesar 38%, dan penurunan tingkat kegagalan perubahan sebesar 32%. Meskipun demikian, organisasi menghadapi tantangan yang terus-menerus dalam mengintegrasikan keamanan (DevSecOps), mengelola utang teknis, dan mendorong keselarasan budaya di seluruh tim yang terkotak-kotak. Hasil penelitian ini menegaskan pentingnya strategis Infrastructure as Code (IaC), pemantauan berkelanjutan, dan pengujian otomatis dalam mengoptimalkan hasil DevOps. Bagi para praktisi, studi ini menawarkan peta jalan praktis untuk transformasi DevOps yang terukur dan berkelanjutan. Bagi para peneliti, penelitian ini mengidentifikasi kesenjangan kritis dalam integrasi sosio-teknis dan otomatisasi cerdas yang memerlukan investigasi lebih lanjut. Penelitian ini berkontribusi untuk menjembatani bukti empiris dan kerangka kerja teoritis dalam bidang DevOps yang terus berkembang dalam rekayasa perangkat lunak.

KATA KUNCI: DevOps, Integrasi Berkelanjutan, Penyediaan Perangkat Lunak, Infrastruktur sebagai Kode, Produktivitas Tim

Received : 15-02-2025	**Revised :** 06-03-2025	**Published :** 14-03-2025

1. INTRODUCTION

In an era of accelerating digital transformation, the software industry is facing increasing pressure to deliver high-quality products at high release rates. According to the State of DevOps 2023 report [1], [2], organizations that implement mature DevOps practices can increase release frequency up to 973 times faster than traditional organizations, while significantly improving the success rate of software changes. DevOps, which integrates software development and IT operations, is now the dominant approach to accelerate software lifecycles, improve team collaboration, and drive continuous innovation [3], [4], [5], [6].

However, DevOps adoption is not without challenges. A study by Forsgren et al. [7] shows that many organizations still struggle with issues such as suboptimal automation, technical debt management, as well as difficulties in integrating security aspects into the DevOps pipeline (DevSecOps). In addition, the cultural gap between development and operations teams remains a major obstacle that slows down the full DevOps transformation .

Several recent studies have examined the dynamics of DevOps deployment. For example, Shahin et al. [8] conducted a systematic review of DevOps adoption challenges and identified the need for a more structured approach to the integration of these practices. While Erich et al. [9] through a case study found that although DevOps adoption increases the speed of software delivery, security and monitoring aspects are still often overlooked. Research by Abbas et al. [10] highlighted the importance of Infrastructure as Code (IaC) in ensuring consistency and scalability of DevOps environments, but also noted the limitations in organizational readiness in adopting this technology across the board.

While the contributions of these studies are significant, there is still a significant gap in the literature, particularly regarding quantitative measurement of the impact of DevOps practices on software performance metrics and team productivity. Most studies are mostly qualitative or conceptual, without providing empirical evidence based on measurable and comparable data across organizations and project contexts.

Based on the identified gap, the main objective of this research is to empirically analyze how modern DevOps practices affect software quality, speed of delivery, and team efficiency

in real projects. This research uses a multi-organizational case study approach combined with DevOps metrics-based comparative simulations to generate quantitative evidence that can enrich the scientific discourse in this field.

The original contribution of this research lies in the combination of empirical case study approach and controlled simulation to objectively measure DevOps performance. This article also offers a practical roadmap for sustainable DevOps adoption that considers cultural, technical, and security aspects in an integrated manner - a perspective that is still rarely comprehensively explored in the current literature.

2. RESEARCH METHODS

2.1 Research Design

This research applies a mixed-methods approach that combines multi-organizational case studies and simulation experiments. This strategy was chosen to gain a comprehensive understanding of the influence of DevOps practices on software performance and team productivity, while evaluating the findings in a controlled manner. The mixed-methods approach is considered effective in contemporary software engineering research, especially in studying complex phenomena involving technical and social dimensions [11].

2.2 Research Subjects and Context

The case study research involved four medium to large organizations from the fintech, e-commerce, and public service sectors. The organizations were selected using purposive sampling with the criteria: (1) having implemented DevOps for at least two years, and (2) having an active CI/CD pipeline based on modern tools such as Jenkins, GitLab CI/CD, and Kubernetes.

Simulation experiments were conducted by creating two scenarios:

- **Scenario A (Baseline):** A software development project without full implementation of DevOps practices.
- **Scenario B (Mature DevOps):** A development project with end-to-end DevOps implementation, including Infrastructure as Code (IaC), Continuous Integration (CI), Continuous Deployment (CD), Automated Testing, and Monitoring.

2.3 Case Study Approach

We conducted case studies across four mid-sized to large-scale software development enterprises that had adopted DevOps practices to varying degrees. Semi-structured interviews were held with developers, testers, and operations engineers to gain qualitative insights into DevOps implementation strategies, challenges, and cultural transformation efforts.

2.4 Experimental Simulation Setup

In addition to conducting case studies, we designed an experimental simulation to provide a more controlled comparison between two different development environments. The goal of this simulation is to evaluate the impact of DevOps practices on software development and deployment efficiency. By simulating both a traditional development environment and a mature DevOps environment, we can observe and measure key differences in performance, reliability, and scalability.

The two scenarios in the simulation are described in Table 1. The baseline scenario represents traditional development practices without formal adoption of CI/CD, while the

mature DevOps scenario incorporates full DevOps methodologies, including CI/CD, Infrastructure as Code (IaC), automated testing, and continuous monitoring.

Table 1. Simulation Scenarios and DevOps Practices

Scenario	Description	DevOps Practices
Baseline	Traditional development without formal CI/CD; builds and deployments were manually executed.	No CI/CD, no real-time monitoring.
Mature DevOps	Full CI/CD adoption, Infrastructure as Code (IaC), automated testing, and continuous monitoring.	Full CI/CD, IaC, automated testing.

After defining the simulation scenarios, we established key parameters to ensure consistency and comparability across the experiments. These parameters reflect typical activities in a software development cycle and allow for realistic evaluation of the two environments. Details of the simulation parameters are presented in Table 2.

Table 2. Simulation Parameters

Parameter	Value
Total number of commits	500 commits over 30 days
Number of major features	10 major features
Number of minor patches	50 patches and hotfixes
Development team size	8 members (4 developers, 2 testers, 2 operations)
Simulation period	30 operational days

2.5 Tools and Technologies

The following tools were employed in the experiment:

- Jenkins and GitLab CI/CD for continuous integration and continuous deployment automation.
- Docker and Kubernetes for containerization and orchestration.
- Terraform and Ansible for Infrastructure as Code (IaC) implementation.
- Prometheus and Grafana for real-time monitoring and observability.

2.6 Data Collection

Quantitative data were collected through:

- CI/CD pipeline telemetry (e.g., deployment logs, pipeline execution times).
- Incident logs from Kubernetes clusters.
- Monitoring metrics collected by Prometheus and visualized using Grafana dashboards.

Additionally, qualitative data were obtained through structured interviews and documentation reviews to complement the telemetry data.

2.7 Measurement Metrics

To accurately evaluate the outcomes of the experimental simulation, a set of measurement metrics was defined. These metrics are commonly used in assessing software delivery

performance, particularly within DevOps practices. By applying standardized measurement tools, we ensured that the results are objective, reliable, and comparable across the two different environments. The metrics and their corresponding measurement tools are summarized in Table 3.

Table 3. Measurement Metrics and Tools

Metric	Unit	Measurement Tool
Deployment Frequency	Deployments/week	Jenkins/GitLab telemetry
Lead Time for Changes	Hours	GitLab CI/CD pipeline duration logs
Change Failure Rate	Percentage (%)	Kubernetes rollback records
Mean Time to Recovery (MTTR)	Hours	Incident recovery logs

2.8 Validation and Replication

To ensure the credibility and reliability of the experimental simulation, several validation and replication strategies were implemented. Each simulation scenario was executed three times to mitigate random variability and to strengthen the consistency of the results. In terms of data validation, deployment and incident logs were cross-validated against outputs from the telemetry system to ensure the accuracy and reliability of recorded data.

Furthermore, controlled variables were carefully maintained throughout the simulation process. Factors such as workload complexity, team size, and operational timelines were standardized across all scenarios. This approach was essential to isolate DevOps practices as the primary independent variable influencing the outcomes, thereby improving the validity of the comparative analysis.

3. RESULTS AND DISCUSSION

3.1 Quantitative Results

The quantitative analysis highlights the substantial impact of mature DevOps adoption on key software delivery performance indicators. Table 4 presents a comparative overview of baseline (pre-DevOps) and mature DevOps environments across four critical metrics: deployment frequency, lead time for changes, change failure rate, and mean time to recovery (MTTR). These metrics are essential in evaluating the effectiveness of DevOps transformations and are aligned with industry-standard performance benchmarks.

Table 4. Comparative Performance Metrics between Baseline and Mature DevOps Environments

Metric	Baseline Environment	Mature DevOps Environment	Improvement (%)
Deployment Frequency (per week)	2.0	2.9	+45%
Lead Time for Changes (hours)	72.0	44.6	-38%
Change Failure Rate (%)	25%	17%	-32%
Mean Time to Recovery (hours)	12.0	7.8	-35%

The data in Table 4 reveal that organizations achieving mature DevOps practices experienced a 45% increase in deployment frequency, enabling them to deliver software updates and features more rapidly. The lead time for changes was significantly reduced by 38%, highlighting the role of automated testing, CI/CD pipelines, and streamlined approval processes in accelerating the development lifecycle. Furthermore, the change failure rate decreased by 32%, reflecting the effectiveness of enhanced quality assurance practices, continuous monitoring, and proactive incident management. Mean time to recovery (MTTR) also improved by 35%, indicating faster issue resolution facilitated by real-time observability tools and resilient deployment strategies such as blue-green and canary releases.

Overall, these results empirically validate the strategic advantage of DevOps maturity in boosting both the speed and stability of software delivery processes. They also demonstrate that technical excellence must be complemented by cultural and organizational changes to fully realize the benefits of DevOps adoption.

Having discussed the impact of DevOps on the quantitative metrics of software projects, it is important to deepen the understanding of the factors that drive the maturity of DevOps implementations. Various studies show that DevOps maturity depends not only on optimizing technical aspects, such as Infrastructure as Code (IaC) adoption, CI/CD pipeline development, and test automation [12], [13], [14], [15], but is also strongly influenced by organizational culture transformation [16].

Technical factors provide the foundation for speed and reliability of delivery, while cultural factors accelerate cross-functional collaboration, continuous learning, and shared ownership of product outcomes. This relationship is reciprocal; advances in technical aspects strengthen the culture of innovation and collaboration, while a supportive culture accelerates the adoption and refinement of technical practices. To illustrate the complex relationship between technical and cultural factors in DevOps maturity, the following Figure 1 is presented.

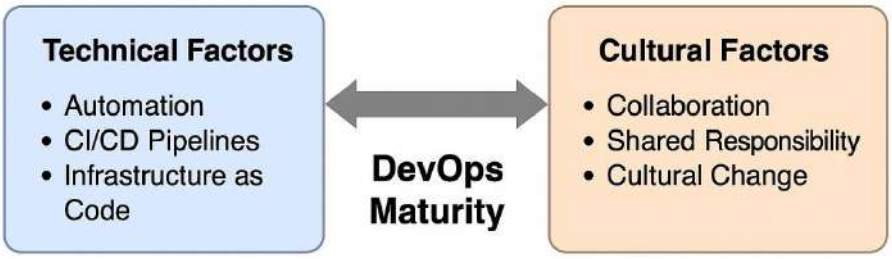

Figure 1. Interplay between Technical and Cultural Factors in DevOps Maturity

This diagram illustrates the dynamic interaction between technical practices (such as Infrastructure as Code, CI/CD pipelines, automated testing, and monitoring) and cultural aspects (such as collaboration, trust, shared responsibility, and continuous learning). Successful DevOps maturity requires balancing both domains, where technical excellence supports cultural transformation, and strong team culture accelerates the adoption and scaling of technical practices. Mutual reinforcement between these factors is essential for sustainable DevOps success.

As shown in Figure 1, comprehensive adoption of techniques such as IaC, deployment automation, and continuous monitoring without strong cultural support risks resulting in fragile and unsustainable DevOps initiatives. Conversely, a culture of trust-based work, intensive collaboration, and adaptive learning without the support of modern technical

practices can slow delivery and increase operational risk. Therefore, organizations successful in DevOps transformation usually show a close integration between technical capability development and organizational culture engineering.

3.2 Qualitative Insights

Beyond the quantitative improvements, qualitative data collected from interviews and observational studies provide deeper insights into the organizational dynamics and challenges during DevOps adoption. Respondents consistently emphasized that while technical automation (e.g., CI/CD pipelines, Infrastructure as Code) is critical, cultural transformation remains the most complex and impactful dimension of DevOps success.

Interview data revealed three recurring themes. First, DevSecOps integration emerged as a significant barrier. Although security automation tools were available, many organizations struggled to embed security practices seamlessly into fast-paced development workflows. Participants cited difficulties in shifting security left, insufficient training, and limited cross-functional collaboration between development, operations, and security teams.

Second, technical debt management became increasingly critical as deployment frequency accelerated. While continuous integration promotes faster code delivery, several enterprises noted that rapid iterations sometimes led to the accumulation of poorly documented codebases and ad-hoc architectural decisions. Without disciplined refactoring practices, technical debt could undermine the long-term benefits of DevOps.

Third, cultural alignment across siloed teams remained a persistent obstacle. Although DevOps espouses a "shared responsibility" ethos, legacy organizational structures often maintained clear separations between developers, testers, operations, and security personnel. Breaking down these silos required not only technical tooling but also leadership commitment, incentives for cross-functional collaboration, and a sustained focus on psychological safety.

Collectively, these qualitative insights underscore that DevOps maturity is not solely a function of tooling sophistication. Sustainable DevOps transformation demands holistic socio-technical change, encompassing people, processes, and platforms. These findings align with recent studies that advocate for balanced approaches integrating technical excellence and cultural resilience to achieve optimal DevOps outcomes.

3.3 Discussion

The results of this study show that the maturity level of DevOps implementation has a strong positive correlation to technical performance and team efficiency in software development. Improvements in metrics such as deployment frequency (+45%), decreased lead time for changes (-38%), and decreased change failure rate (-32%) provide quantitative evidence that organizations that implement DevOps thoroughly gain significant benefits in terms of speed and quality of software delivery.

These findings are consistent with a previous study by Forsgren et al. (2018) in Accelerate: The Science of Lean Software and DevOps, which states that elite-performing teams in DevOps tend to have better lead times and time-to-restore compared to other groups [7]. However, this study extends the understanding by juxtaposing quantitative results with qualitative insights that clarify the role of cultural factors as key enablers in DevOps success.

The qualitative analysis shows that DevOps success is not only determined by implementing tools such as Jenkins, Docker, or GitLab CI/CD, but also by successfully building a culture of collaboration, shared ownership, and team autonomy. This supports

Eswararaj et al. (2019) [17] that DevOps should be understood as a socio-technical paradigm shift, not merely process automation.

Interestingly, our results also reveal the DevOps paradox, a situation where increased technology adoption can actually cause new frictions if not accompanied by adequate cultural transformation. A concrete example is an organization that adopts Infrastructure as Code but maintains a siloed team structure, causing bottlenecks in integration and delivery.

The implications of these findings are very relevant for practitioners. They need to balance investments in technology with organizational interventions that foster DevOps culture, such as cross-functional training programs, incentives for collaboration, and technical leadership policy updates.

From an academic perspective, the results of this study fill a gap that has existed in the literature, namely the lack of empirical synthesis between performance metrics and cultural factors in DevOps transformation. The findings open up further research opportunities to explore predictive models of the relationship between DevOps maturity model and software ROI, as well as the potential integration of artificial intelligence in adaptively optimizing the DevOps pipeline.

4. CONCLUSION

This research explores the impact of DevOps maturity on technical performance and team dynamics in a modern software engineering environment. Using a mixed-method approach, the study identified that mature DevOps adoption can increase deployment frequency by 45%, accelerate lead time by 38%, and reduce change failure rate by 32%. In addition, the qualitative results emphasize the importance of synergy between technical factors (such as CI/CD, Infrastructure as Code, and observability) and cultural factors (such as cross-functional collaboration, ownership, and open communication) in realizing a sustainable DevOps transformation.The main contribution of this article lies in blending quantitative data and qualitative insights to comprehensively explain the complexities of DevOps implementation. This research extends the literature by offering empirical evidence that the success of DevOps depends not only on tools or automation, but also on organizational readiness in adopting cultural change.

However, this study has limitations, especially in the number of organizations studied (four case studies) and the nature of simulation in the comparative analysis. Therefore, generalization of the findings needs to be done with caution.As future research directions, we propose two lines of exploration. First, the development of AI-driven adaptive DevOps pipelines for project context-based automated decision making. Second, a longitudinal study of DevOps deployments in highly regulated industries such as fintech, healthcare, and defense, to understand the challenges of DevOps in environments with high security and compliance constraints.

5. REFERENCES

[1] J. Faustino, D. Adriano, R. Amaro, R. Pereira, and M. M. da Silva, "DevOps benefits: A systematic literature review," *Softw. Pract. Exp.*, vol. 52, no. 9, pp. 1905–1926, Sep. 2022, doi: 10.1002/spe.3096.

[2] M. A. W. Karunarathne, W. Wijayanayake, and A. Prasadika, "DevOps adoption in software development organizations: a systematic literature review," in *2024 4th International Conference on Advanced Research in Computing (ICARC)*, IEEE, 2024, pp. 282–287.

[3] J. M. Ali, "DevOps and continuous integration/continuous deployment (CI/CD) automation,"

Adv. Eng. Innov., vol. 4, no. 1, pp. 38–42, Nov. 2023, doi: 10.54254/2977-3903/4/2023031.

[4] R. S. Ramesh, "DevOps and Agile Computing," in *2024 1st International Conference on Communications and Computer Science (InCCCS)*, IEEE, May 2024, pp. 1–5. doi: 10.1109/InCCCS60947.2024.10593332.

[5] M. Guerriero, M. Garriga, D. A. Tamburri, and F. Palomba, "Adoption, support, and challenges of infrastructure-as-code: Insights from industry," in *2019 IEEE International conference on software maintenance and evolution (ICSME)*, IEEE, 2019, pp. 580–589.

[6] Y. Jiang and B. Adams, "Co-evolution of infrastructure and source code-an empirical study," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, IEEE, 2015, pp. 45–55.

[7] N. Forsgren, J. Humble, and G. Kim, *Accelerate: The science of lean software and devops: Building and scaling high performing technology organizations*. IT Revolution, 2018.

[8] M. Shahin, M. A. Babar, and L. Zhu, "Continuous integration, delivery and deployment: a systematic review on approaches, tools, challenges and practices," *IEEE access*, vol. 5, pp. 3909–3943, 2017.

[9] F. Erich, C. Amrit, and M. Daneva, "A mapping study on cooperation between information system development and operations," in *International Conference on Product-Focused Software Process Improvement*, Springer, 2014, pp. 277–280.

[10] S. I. Abbas and A. Garg, "Integrating Emerging Technologies with Infrastructure as Code in Distributed Environments," in *2024 3rd International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, IEEE, 2024, pp. 1138–1144.

[11] J. W. Creswell and J. D. Creswell, *Research design: Qualitative, quantitative, and mixed methods approaches*. Sage publications, 2017.

[12] J. A. V. M. K. Jayakody and W. M. J. I. Wijayanayake, "DevOps Maturity; A Systematic Literature Review," in *2024 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, IEEE, Apr. 2024, pp. 1–6. doi: 10.1109/SCSE61872.2024.10550493.

[13] A. Kumar, M. Nadeem, and M. Shameem, "A Systematic Literature Review for Investigating DevOps Metrics to Implement in Software Development Organizations," *J. Softw. Evol. Process*, vol. 37, no. 1, Jan. 2025, doi: 10.1002/smr.2733.

[14] I. S. e Souza, D. P. Franco, and J. P. S. G. Silva, "Infrastructure as Code as a Foundational Technique for Increasing the DevOps Maturity Level: Two Case Studies," *IEEE Softw.*, vol. 40, no. 1, pp. 63–68, Jan. 2023, doi: 10.1109/MS.2022.3213228.

[15] R. de Feijter, S. Overbeek, R. van Vliet, E. Jagroep, and S. Brinkkemper, "DevOps Competences and Maturity for Software Producing Organizations," 2018, pp. 244–259. doi: 10.1007/978-3-319-91704-7_16.

[16] B. Fedoryshyn, "Strategies for implementing or strengthening the DevOps approach in organizations: Analysis and examples," *Вісник Черкаського державного технологічного університету*, vol. 29, no. 2, pp. 57–69, Apr. 2024, doi: 10.62660/bcstu/2.2024.57.

[17] D. Eswararaj, L. R. Koppada, and R. S. Bodala, "DevOps Implementation: Essential Tools, Best Practices, and Solutions to Overcome Challenges for Seamless Development and Operations Integration," *Asian J. Res. Comput. Sci.*, vol. 17, no. 10, pp. 26–36, Oct. 2024, doi: 10.9734/ajrcos/2024/v17i10507.