International Journal of Applied Mathematics

Volume 38 No. 11s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

DEVOPS-INTEGRATED QUALITY ENGINEERING FOR CLOUD-BASED PROPERTY AND CASUALTY INSURANCE PLATFORMS

Kiran Babu Boddapati

Acharya Nagarjuna University, Andhra Pradesh, India

shivakiran02@gmail.com

Abstract

The high pace of modernization of Property and Casualty (P&C) insurance delivery has accelerated the adoption of cloud-native architectures and DevOps-based software delivery models. Although these strategies contribute to scalability, agility, and speed-to-market, they are also accompanied by severe difficulties regarding software quality and regulatory compliance as well as operational risk in highly regulated insurance platforms. Increasingly, traditional quality assurance methods, marked by late-stage testing and independent tasks, are not sufficient to deal with the complexity of cloud-based systems, their distributed nature, and constant change. The current paper discusses combining DevOps and Quality Engineering (QE) as a single approach to quality software delivery in cloud-native platforms of P&C insurance. It begins by explaining the weaknesses of traditional insurance IT and quality models in terms of structural constraints that prevent reliability, auditability, and responsiveness. Then, the discussion goes on to discuss the notion of how the principles of DevOps, specifically the idea of continuous integration, automation, observability, and shared ownership can enhance the quality of software by means of continuous validation of the latter integrated into delivery pipelines. It is based on this that this study introduces DevOps-Integrated Quality Engineering (DI-QE) framework to meet the operational and regulatory demands of P&C insurance systems. The framework incorporates automated testing, environment standardization, quality measures, and compliance controls within the continuous delivery work process. It also deals with cloud-native quality engineering, risk-oriented quality indicators and governance mechanisms based on compliance needs of a platform in contemporary insurance. This paper will provide actionable information to the researcher and practitioners interested in enhancing the software reliability, minimizing the delivery risk, and keeping regulatory compliance, harnessing the power of DevOps, and cloud technologies by providing a domain-specific approach to engineering quality into cloud-native insurance systems.

Keywords: Cloud-native Architecture; Continuous Integration; DevOps; Property and Casualty Insurance (P&C); Quality Engineering; Regulatory Compliance.

Introduction

Increased rate of digital transformation is a phenomenon observed in the property and casualty (P&C) insurance sector because of the heightened customer demands, strain on the quality of services owing to the entry of digital-based insurers and the continued changing nature of

Received: Aug 09 2025

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

regulations. Previously, the deployment of core insurance platforms were focused on monolithic and on-premise architecture focusing on stability over a rapid change. However dependable as such systems used to be, they are becoming progressively less able to meet the modern demands e.g. fast release of products, ability to process transactions that can be scaled and periodical update of regulations [1]. Insurers are therefore shifting more to cloud-based and cloud-native systems in the quest to promote operational efficiency, scalability, and agility [1], [2].

Cloud computing offers and enables scalability in the delivery of resources, greater resilience and simple global deployment capacity and is therefore a tactical enabler of massive insurance modernization [1]. However, the distributed systems introduce an architectural and operational complexity of the containerization and service decomposition with cloud adoption. Such modifications may lead to a risk of quality as errors in one integration service may be multiplied and distributed to components that depend on each other, and configuration drift may vary across environments differently [2]. In P&C insurance systems, in which transcription errors, rating error, forms error, and information distortion, and auditability are all zero-tolerance, small flaws can lead to monetary loss, regulatory compliance risk or brand risk.

At the same time, the DevOps is now considered to be a paradigm shift with the emphasis on the strong integration of dev and ops, the high automation, and the feedback loops [3]. Devops and related practices, such as continuous integration, continuous delivery (CI/CD), infrastructure as code and continuous monitoring have been associated with deployment performance and operational outcomes in enterprise systems [4]. Yet, in such controlled spheres as insurance, the speed of delivery cannot take the lead in the DevOps adoption. The quality governance and the risk controls required on the pipelines of delivery should be in a manner that the velocity of the release does not compromise on the reliability, application down time and compliance requirements.

Quality Engineering (QE) is an enhancement of the quality assurance (QA) concept. QE promotes the concept of quality being an end to end engineering practice by removing defects detected at the end of the lifecycle, through automating tests, clear quality gates that are measurable, and continuous lifecycle validation. In conjunction with DevOps, QE can assist with shift-left testing (prevention of bugs early in the design process), which is particularly important to cloud-based P&C platforms where rapid releases are scheduled and any quality failures will be costly [5].

The gap in the paper is proposed to be addressed through the discussion of how both DevOps and Quality Engineering can be smoothly bonded to support the operations of scalable, compliant, and resilient cloud-based platforms of P&C insurance. It gives a point of reference of comparing traditional insurance IT and QA models with those of the cloud-native solutions, evaluates DevOps-driven quality change, and proposes an organized QE framework in accordance with continuous delivery pipelines.

Received: Aug 09 2025

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Cloud Transformation in P&C Insurance Platforms

The move of property and casualty (P&C) insurance platforms to cloud-based platforms is a paradigm shift in the design, deployment and operation of insurance systems. Traditionally, insurance core systems have been created as monolithic apps, highly well-integrated on proprietary infrastructure, slow to deploy, and exhibit poor scale. They were very useful in the case of stationary business requirements but not in the rapidly changing market demands, regulatory needs and the growing customer digital demands [8]. As the digital channels were gaining their presence, and the models of insurance based on usage began to gain prominence alongside the real time risk assessment, insurers began to seek out architectural paradigms, which can help maintain the current change and scalability on demand.

Cloud computing has been one of the sources of this change which provides on demand infrastructure, high availability and even worldwide implementation. With P&C insurers, the modernization of the core business operations of policy administration, claims processing, and billing are facilitated by the service-oriented and microservices-based cloud platforms [9]. Decoupling of the system components allows the insurers themselves to scale critical services and reduces the cost of infrastructure and increases system resilience as well. Other than that, cloud-native architecture allows rapid experimentation and deployment where insurers are able to deploy new products and pricing formulas with much faster speed than they could before with the old systems [10].

The advantages of associating with the cloud in the insurance business are numerous but the high regulatory, security and data control measure complicate the process. The P&C insurance systems expose a great deal of personal and financial data which is highly sensitive and hence compliance to data protection rules, auditability and jurisdiction is essential [11]. As a result, there is a tendency of insurers to follow hybrid or multi-cloud strategies in an attempt to achieve regulatory compliance and the benefits of cloud-elasticity. These strategies introduce additional architectural and functional complexity particularly in the guarantees of homogeneous performance, security and quality in a distributed setting [12].

Cloud transformation also transforms the quality risk profile of insurance platforms in the software engineering perspective. This trend toward distributed systems, container orchestration and automated infrastructure provisioning portends the prospect of configuration mistakes, integration failures and environment specific bugs. The common testing procedures which rely on test environments and late validation is now not useful in this case [13]. Instead, quality must be constantly constructed throughout the development, deployment and operation phases, which justifies the need of integrative practices of DevOps and Quality Engineering.

Altogether, the transformation of organization and functions as the redefinition of software delivery, the risk management, and the quality assurance is not only the transformation of technology platform of P&C insurance to cloud but also the transformation of the organization and functions. This change alone is the only way to put the role of the Quality Engineering developed by DevOps into perspective since it must address the opportunities as much as the threats inherent in its own are posed to cloud-native insurance architectures.

Received: Aug 09 2025

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Traditional Quality Engineering in Insurance Systems

Traditional property and casualty (P&C) insurance systems have long been influenced by monolithic architecture, low release frequency, and highly centralized governance models to pursue quality practices. In such settings, quality engineering has been a downstream practice, commonly known as quality assurance (QA), in which testing was done mainly after the development was done. This method was in line with waterfall or stage-gated delivery models, where requirements, design, implementation, testing, and deployment were considered as consecutive stages [14]. In spite of this sense of control and predictability, this model was increasingly ineffective as the insurance systems were growing in size, complexity and regulatory risk.

Traditional insurance QA processes were based on manual test execution and regression testing in a closed environment and document-based validation. Test cycles were usually long and resource-intensive so that it was hard to support the high rate of business rule changes like the logic of ratings, underwriting requirements or regulatory requirements. Consequently, quality validation was frequently a queue in the release pipelines, which strengthened long delivery lead times and inhibited innovation [15]. More so, the use of late stage testing implied that any defects that occurred in the initial stages, either in terms of architecture or incorporation, were not found on time thus creating a much more expensive remediation cost.

Environment dependency is another drawback of traditional quality engineering in insurance systems. The legacy platforms were usually deployed to a limited number of tightly regulated environments like the development, system testing, user acceptance testing and the production. These environments were costly to supply and challenging to recreate which resulted in a mismatch between the test and production environments [16]. These environment gaps in P&C insurance platforms can often lead to production incidents that were not identified during testing, given that the volume of transaction, the amount of concurrency, and the variability of data can be high.

The traditional models of QA have also reinforced the silos within an organization, with respect to the governance. The development team, testing team, and the operations team typically worked independently without much collective responsibility towards quality deliverables. This kind of segregation curtailed feedback mechanisms and reduced the ability to learn based on the production errors or operational failures [17]. The siloed organization also complicated the auditability and traceability in the regulated insurance circumstances as quality evidence was often divided between the teams and tools.

The flaws in conventional quality engineering were more apparent once the insurance platforms started switching to cloud-based and service-oriented designs. Continuous validation and quick feedback is required by distributed systems, frequent releases, and automated infrastructure provisioning, but traditional models of QA are not meant to provide them [18]. Such constraints necessitate the necessity to transition away to more progressive QA practices and adopt Quality Engineering practices that incorporate quality into the software lifecycle.

Received: Aug 09 2025

Volume 38 No. 11s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

This development preconditions the introduction of Quality Engineering that is incorporated into DevOps and discussed later.

DevOps Principles and Their Impact on Software Quality

DevOps has become a radical paradigm of software engineering, with a fundamental restructuring of the process of quality provision and maintenance of software in the contemporary systems. DevOps encourages a socio-technical way of development, testing, and operations as opposed to separate processes that assume shared accountability, automation, and continuous feedback throughout the software life-cycle. In the simplest form, DevOps is intended to alleviate delivery friction and enhance system reliability and quality by applying continuous integration, continuous delivery (CI/CD), automated testing, infrastructure as code, and continuous monitoring [19].

The importance of DevOps on the quality of software has been one of the most important aspects as it focuses on early and consistent validation. Continuous integration motivates developers to incorporate changes into code regularly in order to identify defects and integration problems very quickly. Unit, integration, and system-level tests are performed automatically as part of the pipeline, making sure that quality checks are always enforced with each change [20]. The shift-left strategy minimizes the defect leakage to the subsequent stages and minimizes the cost of fixing bugs, which is critical with complex, cloud-based systems.

Besides early validation, DevOps introduces the notion of shift-right quality, which introduces quality engineering into the production environment. Real-time monitoring, recording, and flagging provide data on the system behaviour and performance as well as failure mode in the actual use conditions. The teams can identify the hidden defects, performance defects and bug reliability that would be not otherwise observable in the testing of pre-production process due to such feedback loops [21]. Such operational visibility may allow managing incidents much faster with auditability and traceability being preserved in controlled spheres such as insurance.

Another principle of DevOps that has a direct implication on quality is automation. IaC and automated provisioning of environments minimizes configuration drift, and provides consistency in the development, testing, Preproduction and production environments. This uniformity is essential to reduce the environment-specific defects, which are typical of distributed and cloud-native systems [22]. In addition, automated deployment pipelines impose standardized quality gates, and quality assurance is an inseparable aspect of the delivery process and not an external checkpoint.

Nevertheless, DevOps does not dissolve the disciplined quality engineering, but rather increases it. DevOps pipelines have the potential to jumpstart the spread of defects without well-established quality metrics, governance systems, and risk controls. As a result, the success of DevOps relies on the incorporation of quality goals, i.e., reliability, security, and maintainability, into pipeline design and practices of the team directly [23]. This has led to the development of DevOps-based Quality Engineering, where quality is seen as an on-going engineering issue, and not a phase of testing.

Received: Aug 09 2025

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Cloud-Native Quality Engineering Architecture for P&C Insurance Platforms

A cloud-native architectural backbone is necessary to implement the concept of DevOps-Integrated Quality Engineering (DI-QE) in property and casualty (P&C) insurance platforms to facilitate automation and scaling as well as sustained validation. The cloud-native insurance platforms are of services loosely coupled, dynamically provisioned, continuous deployment pipelines unlike the traditional centralized systems. The engineering of quality in such environments should then be designed as a cross cutting and an architectural ability and not as an abstraction testing layer [29].

A cloud-based QE design will normally assume a layer-based model. The first layer is the infrastructure layer, in which infrastructure as code allows the provisioning of similar and reproducible environments in development, testing, and production. This architectural concept minimizes configuration drift and makes quality checks to be performed in environments that are as close as possible to actual operation environments. This alignment is critical in insurance platforms where price correctness, transactions consistency and reporting required by regulations, all depend on environmental correctness [30]. Automated environment provisioning is also necessary to provide on demand test environments because it enables parallel testing strategies and faster test environment feedback.

CI/CD pipeline is the implementation base of quality engineering that is placed above the infrastructure layer. Continued integration of pipelines that are constantly tested are the pipelines that execute automated testing and code static analysis, security testing and compliance testing on every code change. The pipelines are also used as quality enforcement systems since they incorporate quality gates which must be left to proceed with the deployment process. These gates are required in cloud-based P&C platforms to authorize the sophisticated business logic, inter-point services between policy lifecycle services, and inter-component information consistency [31].

Observability is another QE architectural dimension requirement of a cloud-native. Monitoring, logging and tracking systems provide real time data on application and service interaction behavior in the production process. This observability data will aid in the implementation of shift-right quality since this data will aid in establishing the performance bottlenecks, fault patterns, and operational anomalies that are not readily observed in pre-production environments. The feedback of the production system assists in designing the tests and quality standards that define a continuous learning process that enhances resilience of the system [32].

Finally, cloud-native QE architectures can also be described as being scalable and fault tolerant. There are distributed testing strategies, service virtualization and contract testing that is applied to test service interactions without complete system deployments. The approach can be of particular use in situations involving insurance ecosystems of large scale, in which case it is common to be dependent on external partners, regulatory frameworks and third party data providers. In the case of the cloud-native principles, the consistency between the quality engineering architecture assists the insurers to proceed with the high delivery rates without

Received: Aug 09 2025

Volume 38 No. 11s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

reducing the high-quality levels of reliability and compliance demanded by the P&C insurance applications.

Quality Metrics, Compliance, and Risk Management in P&C Insurance Systems

The cloud-based property and casualty (P&C) insurance platforms quality engineering goes far beyond the functional correctness and encompasses regulatory compliance, operational risk management, and system resilience. Unlike most other tasks within an enterprise, the insurance systems are very regulated and must be able to be audited, traced, and data integrity ensured throughout the entire software delivery chain. As insurers switch to DevOps and cloud-native architectures, quality measurements will also be necessary to measure both compliance assurance and technical performance simultaneously [33].

Classical quality indicators, of defect rate or test results, provide only mediocre data on the dependability and regulatory readiness of distributed systems. The original P&C services on clouds need multidimensional quality indicators that are indicative of availability, latency, fault-tolerance, security posture and recoverability. More and more metrics are also applied in order to quantify quality of delivery and metrics of operational risk such as service-level objectives (SLOs), error budgets, mean-time-to-recovery (MTTR), and change failure rates [34]. These measures enable companies to exchange deployment speed with stability which is one of the primary issues of insurance platforms that operate with high-value financial transactions and customer sensitive data.

The other aspect that defines quality engineering in P&C systems through compliance consideration is the manner in which it is implemented. The regulatory frameworks typically involve the fact that the insurers must have a clear traceability among the business requirements, in place controls, test evidence and production behaviour. To address the needs, Quality Engineering reinforced by DevOps might aid in automating the compliance control, policy validation, and evidence generation, which could be directly incorporated in CI/CD pipelines. This capability to access automated logging, audit trail and have infrastructure configurations capable of version control can help insurers demonstrate compliance without necessarily incurring manual documentation processes which are susceptible to errors as well as being difficult to scale [35].

The risk management is closely related with the quality engineering in the insurance systems. The financial loss and regulatory fines can be directly converted into software bugs and performance issues or even security threats. Cloud-native QE systems can mitigate such risks to make sure that they are minimized with the help of continuous monitoring, anomaly detection, and controlled release mechanisms such as canary deployment and automated rollback. The techniques reduce the radius of blast of the failures and promote proactive reduction of risk rather than reactive response to the incidents [36].

In general, quality metrics, compliance control, and risk management mechanisms in the cloud-based P and C insurance platforms must be set in such a way that they must be the functioning ones in a set. The structural and operational foundation that needs to be included in the DevOps

Volume 38 No. 11s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

is Quality Engineering, which is needed to coordinate the fast delivery with regulatory responsibility and enable the insurers to be innovative without reducing trust, stability and compliance.

Conclusion

The replacement of property and casualty (P&C) insurance systems by cloud-native systems has altered radically the manner in which software systems are designed, implemented and managed. Despite the fact that cloud and Devops practices may guarantee scalability, agility, and quick innovation, cloud and Devops introduce new quality, compliance, and operational risks which cannot be successfully addressed using the conventional quality assurance strategies. The paper has discussed these problems in the context of DevOps-enabled Quality Engineering (DI-QE), which puts quality as a continuous shared engineering task instead of a downstream verification process.

The study set a distinct base of the traditional insurance IT and QA practices, which helped to emphasize a set of structural constraints that obstruct the quality assurance of modern, distributed environment. It has then shown how the principles of DevOps, coupled with the systematic practice of Quality Engineering, allow continuous validation, consistency of the environment, feedback-based on observability, and delivery awareness to risk. The suggested DI-QE framework directly places automated testing, quality indicators, and compliance controls in the same step with the CI/CD pipelines, so that the quality and regulatory requirements can change in pace with the high-delivery frequency.

To P&C insurers, quality engineering combined with DevOps is not a technical optimization, it is a strategic requirement. Sound policy processing, sound claims processing and safe data management are part of organizational trust and regulation compliance. DI-QE offers a well-organized channel of realizing such goals and maintaining innovation and operational flexibility. Since insurers are still modernizing their systems, the future research and practice needs to be oriented at the further development of autonomous quality management, AI-controlled testing, and automated compliance to further improve the quality results in the cloud insurance ecosystems.

References

[1] Marston, S., Li, Z., Bandyopadhyay, S., Zhang, J., & Ghalsasi, A. (2011). Cloud computing—The business perspective. Decision Support Systems, 51(1), 176–189. https://doi.org/10.1016/j.dss.2010.12.006

[2] Pahl, C. (2015). Containerization and the PaaS cloud. IEEE Cloud Computing, 2(3), 24–31. https://doi.org/10.1109/MCC.2015.51

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

Received: Aug 09 2025

Volume 38 No. 11s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[4] Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and DevOps: Building and scaling high performing technology organizations. IT Revolution. ISBN: 9781942788331. IT Revolution+1

[5] Leite, L., Rocha, C., Kon, F., Milojicic, D., & Meirelles, P. (2019). A survey of DevOps concepts and challenges. ACM Computing Surveys, 52(6), Article 127. https://doi.org/10.1145/3359981

[6] Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation. Addison-Wesley Professional. ISBN: 9780321601919. Pearson+1

[7] Newman, S. (2021). Building microservices: Designing fine-grained systems (2nd ed.). O'Reilly Media. ISBN: 9781492034025.

[8] Janssen, Marijn & Joha, Anton. (2011). Challenges for adopting cloud-based software as a service (SAAS) in the public sector. 19th European Conference on Information Systems, ECIS 2011.

[9] Villamizar, M., Garcés, O., Castro, H., Salamanca, L., Casallas, R., & Gil, S. (2016). Evaluating the monolithic and the microservice architecture pattern to deploy web applications in the cloud. IEEE Computing Conference, 583–590. https://doi.org/10.1109/SAI.2016.7556016

[10] Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. Journal of Systems and Software, 150, 77–97. https://doi.org/10.1016/j.jss.2019.01.001

[11] Singh, S., Jeong, Y.-S., & Park, J. H. (2016). A survey on cloud computing security: Issues, threats, and solutions. Journal of Network and Computer Applications, 75, 200–222. https://doi.org/10.1016/j.jnca.2016.09.002

[12] Pahl, C., Brogi, A., Soldani, J., & Jamshidi, P. (2019). Cloud container technologies: A state-of-the-art review. IEEE Transactions on Cloud Computing, 7(3), 677–692. https://doi.org/10.1109/TCC.2017.2702586

[13] Dragoni, N., Giallorenzo, S., Lluch Lafuente, A., Mazzara, M., Montesi, F., Mustafin, R., & Safina, L. (2017). Microservices: Yesterday, today, and tomorrow. Present and Ulterior Software Engineering, 195–216. https://doi.org/10.1007/978-3-319-67425-4_12

[14] Boehm, B. W. (1988). A spiral model of software development and enhancement. Computer, 21(5), 61–72. https://doi.org/10.1109/2.59

[15] Garousi, V., Felderer, M., & Mäntylä, M. V. (2019). Guidelines for including grey literature and conducting multivocal literature reviews in software engineering. Information and Software Technology, 106, 101–121. https://doi.org/10.1016/j.infsof.2018.09.006

Received: Aug 09 2025

Volume 38 No. 11s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[16] Hossain, M. A., Jiang, J., Han, J., Kabir, M. A., Schneider, J.-G., & Liu, C. (2022). Inferring data model from service interactions for response generation in service virtualization. Information and Software Technology, 145, 106803. https://doi.org/10.1016/j.infsof.2021.106803

[17] Catal, C., & Diri, B. (2009). A systematic review of software fault prediction studies. Expert Systems with Applications, 36(4), 7346–7354. https://doi.org/10.1016/j.eswa.2008.10.027

[18] Chen, L. (2015). Continuous delivery: Huge benefits, but challenges too. IEEE Software, 32(2), 50–54. https://doi.org/10.1109/MS.2015.27

[19] Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. IEEE Software, 33(3), 94–100. https://doi.org/10.1109/MS.2016.68

[20] Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. Journal of Systems and Software, 128, 85–100. https://doi.org/10.1016/j.jss.2017.03.048

[21] Fitzgerald, B., & Stol, K.-J. (2017). Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176–189. https://doi.org/10.1016/j.jss.2015.06.063

[22] Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. Computer Science Review, 38, 100308. https://doi.org/10.1016/j.cosrev.2020.100308

[23] Zhou, X., Mao, R., Zhang, H., Dai, Q., Huang, H., Shen, H., Li, J., & Rong, G. (2023). Revisit security in the era of DevOps: An evidence-based inquiry into DevSecOps industry. IET Software, 17(4), 435–454. https://doi.org/10.1049/sfw2.12132

[24] Mishra, A., & Otaiwi, Z. (2020). DevOps and software quality: A systematic mapping. Computer Science Review, 38, 100308. https://doi.org/10.1016/j.cosrev.2020.100308

[25] Lenarduzzi, V., Lomio, F., Saarimäki, N., & Taibi, D. (2020). Does migrating a monolithic system to microservices decrease the technical debt? Journal of Systems and Software, 169, 110710. https://doi.org/10.1016/j.jss.2020.110710

[26] Bass, L., Weber, I., & Zhu, L. (2015). DevOps: A software architect's perspective. Addison-Wesley Professional. ISBN: 9780134049847

[27] British Standards Institution. (2011). BS ISO/IEC 25010:2011—Systems and software engineering. Systems and software quality requirements and evaluation (SQuaRE). System and software quality models. BSI Standards Limited. https://doi.org/10.3403/30215101

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

Received: Aug 09 2025

Volume 38 No. 11s, 2025

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[29] Santos, M. A., & Ribeiro, C. (2018). A systematic mapping study on microservices. In Communications in Computer and Information Science (pp. 283–294). Springer. https://doi.org/10.1007/978-3-319-99007-1_100

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

[31] Hilton, M., Tunnell, T., Huang, K., Marinov, D., & Dig, D. (2016). Usage, costs, and benefits of continuous integration in open-source projects. In Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE '16) (pp. 426–437). ACM. https://doi.org/10.1145/2970276.2970358

[32] Li, B., Peng, X., Xiang, Q., Wang, H., Xie, T., Sun, J., & Liu, X. (2022). Enjoy your observability: An industrial survey of microservice tracing and analysis. Empirical Software Engineering, 27(1), Article 25. https://doi.org/10.1007/s10664-021-10063-9

[33] Zadeh, A., Lavine, B., Zolbanin, H., & Hopkins, D. (2023). A cybersecurity risk quantification and classification framework for informed risk mitigation decisions. Decision Analytics Journal, 8, 100328. https://doi.org/10.1016/j.dajour.2023.100328

[34] Kersten, A., & van der Wal, J. (2021). Measuring software delivery performance using the four key metrics of DevOps. In P. Gregory et al. (Eds.), Agile Processes in Software Engineering and Extreme Programming (XP 2021) (Lecture Notes in Business Information Processing, Vol. 419, pp. 103–119). Springer. https://doi.org/10.1007/978-3-030-78098-2_7

[35] Savor, T., Douglas, M., Gentili, M., Williams, L., Beck, K., & Stumm, M. (2016). Continuous deployment at Facebook and OANDA. In Proceedings of the 38th International Conference on Software Engineering Companion (ICSE '16 Companion) (pp. 21–30). ACM. https://doi.org/10.1145/2889160.2889223

[36] Taibi, D., Lenarduzzi, V., & Pahl, C. (2017). Processes, motivations, and issues for migrating to microservices architectures: An empirical investigation. IEEE Cloud Computing, 4(5), 22–32. https://doi.org/10.1109/MCC.2017.4250931

Received: Aug 09 2025