IJSES
International Journal of Scientific Engineering and Science
Volume 10, Issue 6, pp. 46-51, 2026.
ISSN (Online): 2456-7361

Impact of Continuous Delivery Frequency on Regression Reduction and Feedback Efficiency in Large-Scale Software Systems

Evgenii Lvov
Head of engineering, senior full stack architect, Batumi, Georgia

Abstract— *The paper presents an extensive theoretical and empirical study of the relationship between the frequency of Continuous Delivery (CD), the effectiveness of regression test reduction, and the dynamics of the feedback loop in large-scale software systems (Large-Scale Software Systems). Drawing on statistical analysis of the State of DevOps 2024 data in combination with an in-depth examination of the practices of leading technology companies (Google, Meta, Spotify), the author demonstrates that a straightforward increase in deployment frequency in the absence of intelligent test selection mechanisms (TIA) leads to an exponential increase in the Rework Rate metric and a degradation of overall system stability. The paper introduces the concept of the cognitive cost of feedback, which makes it possible to quantify the load on engineering teams under an accelerated delivery cycle, and substantiates that the most mature teams achieve high development throughput not by maximizing test coverage, but by using probabilistic failure prediction models and automated filtering of unstable tests. The conclusions formulated in the paper lead to an empirically validated model of the transition from strictly deterministic testing approaches to stochastically oriented quality management in highly complex software and hardware ecosystems.*

Keywords— *Continuous Delivery, Regression Testing, Test Impact Analysis, Rework Rate, Feedback Efficiency, Large-Scale Systems, DORA Metrics, Flaky Tests, Transition Prediction, DevOps Evolution.*

I. INTRODUCTION

The modern software development industry is undergoing a profound structural transformation associated with the widespread transition to hyperscale architectures and intensifying competition for reducing the product time-to-market cycle. The paradigm of continuous delivery, which asserts the necessity of frequent, predictable, and reliable releases, has established itself as the de facto standard for technologically mature organizations. Nevertheless, as systems evolve from monolithic applications to distributed microservice landscapes and monorepositories with millions of lines of code, classical quality assurance practices begin to demonstrate limited scalability and increasing maintenance cost [1].

By 2024, this tension has only intensified: the inertial increase in delivery frequency without synchronized optimization of verification procedures has led to a noticeable deterioration in stability indicators. According to long-term observations by DevOps Research and Assessment (DORA), for the first time in the last decade there has been an increase in the share of low-performing teams (Low Performers) from 17% in 2023 to 25% in 2024 [3]. This change might be interpreted as a sign that the sector has reached the limit of accelerating delivery speed: further progress is made possible mostly by drastically enhancing the efficiency of internal processes, which includes cutting down on the amount of regression testing while maintaining the necessary coverage. In this context, the relevance of the study is determined by the need for a theoretical and practical understanding of mechanisms that make it possible to sustain a high rate of change while simultaneously minimizing feedback cycle time, which is critically important for preventing the accumulation of technical debt and reducing the cognitive load on developers caused by constant context switching and processing of noisy or delayed quality information.

An analysis of the state of the software development industry in 2024, based on empirical data from DORA and DX reports, reveals a number of macrotrends that require in-depth interpretation. First, there is an erosion of the middle class of highly effective teams: the share of the High Performers cluster decreased from 31% to 22%, which indicates the growing difficulty of maintaining high performance standards under conditions of increasing architectural complexity of systems and integration of components based on artificial intelligence [3]. This shift reflects a systemic difficulty in scaling DevOps practices in parallel with the growth in code volumes, the number of interdependent services, and reliability requirements. Second, in 2024 the Rework Rate metric was introduced into the DORA analytical model, characterizing the share of time spent on rework and defect correction; empirical studies demonstrate a pronounced correlation between Change Failure Rate (CFR) and Rework Rate, which confirms the hypothesis that unstable and poorly controlled delivery processes generate toxic workload that crowds out innovative and exploratory activities of teams [4]. In other words, the growth of CFR leads to a reallocation of engineering time in favor of reactive elimination of the consequences of unsuccessful changes.

An additional destabilizing factor is the phenomenon of flaky tests. In web-scale systems, a typical example of which is the Google infrastructure with about 4.2 million tests, from 1.5% to 2% of automated tests exhibit unstable, flaky behavior. At a high run frequency within Continuous Integration pipelines, this level of flakiness generates a significant number of false positives, which leads to blocking up to 16% of runs not associated with real defects [6]. As a result, the test infrastructure ceases to be perceived as a reliable source of quality signals and turns into a noise generator, undermining

confidence in CI/CD pipelines, slowing down releases, and increasing the cognitive and emotional pressure on engineering teams.

At the same time, the existing body of scientific and applied literature, on the one hand, describes in sufficient detail the advantages of continuous delivery,[1] and on the other hand, pays significant attention to algorithmic aspects of test selection (Regression Test Selection, RTS) [7]. Nevertheless, there remains a pronounced deficit of comprehensive studies that directly link delivery frequency with the effectiveness of regression test reduction through the lens of the new Rework Rate metric under conditions of real industrial scale. Most academic works rely on isolated experiments with open-source projects that often do not reproduce the characteristic features of industrial monorepositories: extreme scale, high density of interdependencies, complex organizational structures, and regulatory constraints. As a result, there is no holistic model describing how intelligent test selection (TIA) affects not only formal engineering indicators (Developer Productivity) but also the psychological aspects of developers' interaction with the feedback system, including the phenomenon of feedback fatigue.

Under these conditions, the research objective is formed as the development of a theoretical and empirical model of the impact of continuous delivery frequency on the effectiveness of regression test reduction methods and on the speed of the feedback cycle, as well as the identification of such modes and organizational and technical conditions under which maximizing delivery speed does not lead to an increase in Rework Rate.

The scientific novelty of the work lies in the development of a theoretical and empirical model of the impact of continuous delivery frequency on Rework Rate and the effectiveness of the feedback loop in large-scale software systems, with the introduction of the concept of cognitive cost of feedback, demonstrating that sustainable reduction of regression is achieved not through extensive growth of test coverage, but through the implementation of stochastically oriented test selection methods (RTS/TIA) and systematic management of flaky tests.

The first research idea is that there is a limiting value of effective delivery frequency in large-scale software systems. Beyond this value, additional linear acceleration of releases without the use of probabilistic methods of test reduction (RTS/TIA) causes a nonlinear increase in Rework Rate and a decrease in feedback time. This effect is caused by the cumulative influence of false positive results (flaky failures), which intensify the cognitive overload of developers and reduce their ability to promptly interpret quality signals and make well-founded engineering decisions.

II. Materials and Methods

This study employs a mixed research approach, within which quantitative analysis of industrial metrics is combined with qualitative interpretation of architectural and organizational patterns. The quantitative component is implemented in the form of comparative statistical analysis based on aggregated data from industry reports DORA 2024, DX, and RedMonk, used for clustering engineering teams by performance levels. To identify relationships between throughput and stability, correlation analysis methods were applied, which made it possible to determine how changes in delivery frequency are reflected in process robustness and release quality. The qualitative component is represented by a series of in-depth case studies focused on companies operating ultra-large-scale systems. In particular, for Google the evolution of the Test Automation Platform (TAP) and the integration of Transition Prediction algorithms aimed at reducing the impact of unstable tests and filtering noisy triggers were analyzed. For Meta (Facebook) the subject of study was the Conveyor continuous deployment system and the use of TestGen-LLM for generating mutation tests, which reflects the advanced level of application of generative models (Generative AI) in the field of testing. In the case of Spotify, the impact of introducing the internal developer portal Backstage and Test Impact Analysis (TIA) practices on team performance was examined under conditions of optimization and reallocation of limited computational and organizational resources. Additionally, a systematic literature review was conducted on publications in the IEEE Xplore and ACM Digital Library databases in order to identify theoretical foundations and algorithmic approaches to test selection methods, including RTS and TIA.

III. Results and Discussion

The analysis of metrics for 2024 captures a qualitative rethinking of the DevOps performance category itself. While the prevailing thesis in the industry previously stated that increased delivery speed does not conflict with quality, current data demonstrate the presence of a hidden parameter, Rework Rate, which in fact determines the true effectiveness of development and operations processes. To put it another way, if rework and the number of reactive operations are not considered, a nominal increase in release frequency is no longer a useful indicator of efficiency and may hide systemic instability in the process [9, 12].

Against this background, the polarization of the distribution of teams by performance level becomes particularly evident. The gap between the Elite and Low clusters is widening: elite teams (approximately the top 5–10%) already rely on an on-demand deployment model, achieving a time to restore service of less than one hour, whereas teams from the Low cluster demonstrate regressive dynamics: deployment frequency decreases to a few releases per month, and the Change Failure Rate (CFR) reaches the range of 40–60%.[20] At the same time, the anomaly of the Medium cluster is particularly noteworthy, as in 2024 it exhibits a lower CFR compared to the High cluster.[10] Such a shift disrupts the expected linear relationship in which higher delivery speed should be accompanied by stable or controllable failure indicators. This divergence can be interpreted as a manifestation of a complexity barrier: when transitioning from moderate to high delivery speed, teams encounter a situation in which traditional approaches to testing and quality control no longer scale, while new methods—primarily automated TIA practices and the associated test selection algorithms—have not yet been

implemented to a sufficient degree to offset the increased load on the quality assurance system.

Table 1 presents a comparative performance metric matrix by clusters.

TABLE 1. Comparative matrix of performance metrics by clusters (compiled by the author based on [4, 5, 21]).

Metric	Elite Performers	High Performers	Medium Performers	Low Performers	2024 dynamics
Deployment Frequency	On demand (On-demand)	Daily – Weekly	Weekly – Monthly	Once per month – once every 6 months	Polarization has intensified
Lead Time for Changes	< 1 hour	1 day – 1 week	1 week – 1 month	1 – 6 months	Stagnation in the Low cluster
Change Failure Rate	~5%	10% – 15%	10% – 15% (anomaly)	40% – 64%	Correlation with Rework Rate
Failed Deployment Recovery	< 1 hour	< 1 day	1 day – 1 week	1 week – 1 month	Moved to the Throughput category
Rework Rate	Minimal (<10%)	Moderate	High	Critical (>30%)	New stability metric

The introduction of the Rework Rate metric has made it possible to quantitatively assess the cost of speed in modern DevOps processes: it has become clear that a high release frequency by itself does not guarantee efficiency if a significant share of effort is spent on rework. Elevated Rework Rate values observed in the Low and Medium clusters indicate that a substantial portion of throughput is effectively wasted. Empirical studies show a stable direct correlation: teams with a high Change Failure Rate (CFR) almost inevitably exhibit a high Rework Rate, forming a vicious loop in which bug fixing, performed under conditions of haste and limited regression testing, becomes a source of new defects [5, 18].

In the context of large-scale systems comparable in size to the infrastructures of Google or Meta, the Retest All strategy for every commit is practically infeasible: a full execution of the test suite is constrained both by a combinatorial explosion of states and by unacceptable execution time. Under such conditions, the efficiency of the delivery pipeline is directly determined by the quality and sophistication of test selection algorithms, that is, by the ability of the system to choose a minimal yet sufficient subset of tests relevant to a specific change.

In this context, Test Impact Analysis (TIA) serves as an evolutionary advancement of the entire regression testing paradigm rather than merely a specific optimization approach. In contrast to approaches based solely on static dependency analysis, modern TIA systems rely on dynamic call graphs, runtime execution data, and code change history. The results of studies conducted on industrial software systems demonstrate that combining TIA with Pareto-oriented testing techniques makes it possible to reduce the total test execution time while preserving error detection capability (Fault Detection Capability) [19]. A practical example is provided by Spotify, where the introduction of TIA within the Backstage platform made it possible to reduce computational resource consumption by approximately 50%, which became a critical factor for operational sustainability (Sustainability) and for reducing the carbon footprint of the infrastructure [15]. Thus, TIA serves not only as a mechanism for accelerating feedback but also as a tool for optimizing operational expenditures in the logic of FinOps.

The Google case illustrates how intelligent test selection can be used to combat the entropy of a massive monorepository. In a setting where a single commit can potentially affect thousands of dependent modules, the company relies on the TAP (Test Automation Platform) system, within which the Transition Prediction algorithm is implemented [11]. The essence of this approach lies in analyzing the history of test runs and predicting the probability that a particular test will transition from a PASS state to a FAIL state under the influence of the current change. Tests with an extremely low probability of such a transition — as a rule, those that are consistently passing and not directly related to the modified code regions — are excluded from the blocking pre-submit stage and moved to post-submit. This makes it possible to radically reduce the time of critical feedback — from hours to minutes — while maintaining high confidence that the most risky regressions will not go undetected [17, 22].

Meta, however, exhibits a distinct yet congruent development path, depending on highly regular continuous deployment throughout the Conveyor system [13]. The key innovation of 2024 was the introduction of TestGen-LLM, a test generation system based on large language models [14]. This system implements a mutation-guided approach: the generated tests are purposefully aimed at detecting regressions in the specifically modified code regions rather than at abstractly increasing coverage. An important effect is the semantic awareness of the LLM: the model is capable of interpreting the meaning of a change and generating tests with high substantive relevance and a reduced rate of false positives compared to traditional generators. This, in turn, makes it possible to keep the Rework Rate at a relatively low level even with thousands of daily deployments, without degrading the perceived quality of feedback.

The speed and predictability of feedback become a central parameter of process architecture. In accordance with Little's law, an increase in testing time under a constant rate of incoming commits inevitably leads to the growth of the task queue, the accumulation of work in progress, and more frequent context switching. For developers this means not only an increase in calendar time to defect detection, but also a deterioration in the perceived quality of the signal: by the time the result is obtained, the connection between the change and its effect is often already blurred.

Against this background, unstable tests, that is, tests capable of producing different results on an identical version of the code, act as one of the key destabilizing factors of Feedback Efficiency. In Google's infrastructure, about 1.5% of all automated tests exhibit flaky behavior, and at a scale of about

4.2 million tests this generates a colossal volume of noisy failures [6]. Studies conducted in 2024 show that even a relatively small percentage of flaky tests substantially undermines developers' trust in the CI system: if a red build with a probability on the order of 50% is caused by an infrastructural or flaky error, developers gradually stop responding to such signals, which leads to real defects being missed and, as a direct consequence, to an increase in Rework Rate. To mitigate this effect, Flake Aware Culprit Finding strategies are used; they automatically rerun failed tests, statistically confirm or refute their instability and, in the event that flakiness is detected, isolate such tests without blocking the release [11].

For additional verification of these observations, a visual model of the correlation between failure frequency and rework volume is constructed on the basis of DORA data, demonstrating that an increase in CFR is systematically associated with an increase in Rework Rate and, consequently, with a reduction in the share of useful Throughput that is actually directed toward creating new functionality rather than eliminating the consequences of unstable delivery.

The correlation between CFR and Rework Rate by clusters is presented below in Figure 1.

The chart shown in Figure 1 demonstrates a positive correlation between change failure rate (CFR) and the level of rework. The Low cluster is characterized by critically high values of both metrics, which confirms the hypothesis of process inefficiency. A comparison of the traditional Retest All approach and TIA shows a non-linear benefit under scaling (see Fig. 2).

Figure 2 demonstrates that for typical commits affecting less than 10% of the system, resource savings reach 80–90%. The advantage disappears only in the case of global architectural changes.

Table 2 presents the results of a comparative analysis of the effectiveness of regression testing strategies.

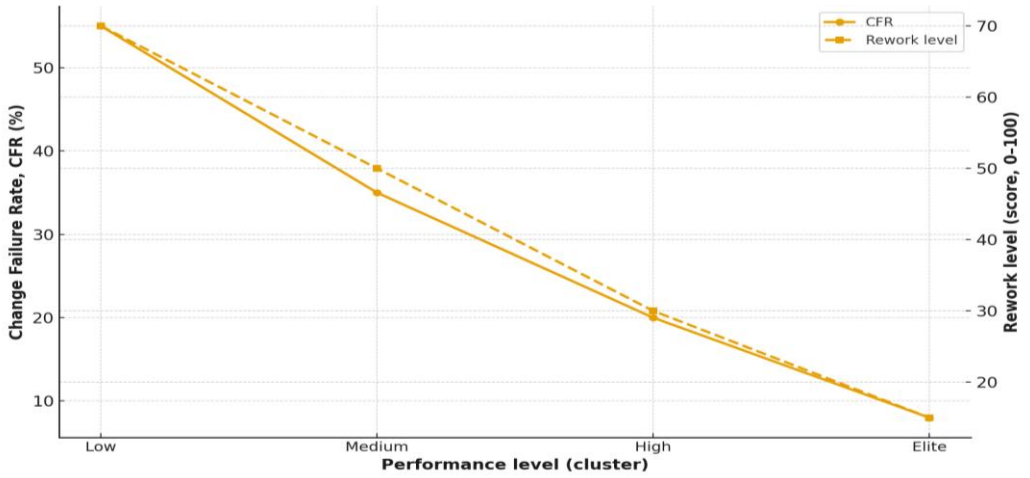

Fig. 1. Correlation between CFR and Rework Rate by clusters (compiled by the author based on [6, 11, 13]).

Fig. 2. Effectiveness of TIA vs. Retest All depending on the volume of changes (compiled by the author based on [6, 11, 13]).

Despite the empirically validated effectiveness of advanced continuous delivery strategies, their practical adoption in traditional enterprise environments encounters a set of systemic barriers.

First, cultural inertia plays a significant role: the historically established separation of responsibilities between development, operations, and testing teams (the classical Dev vs Ops model) reinforces the absence of unified quality metrics and shared accountability for outcomes. This makes the integration of end-to-end CD practices difficult, since each functional area

optimizes local indicators rather than global reliability and the throughput of value delivery.

TABLE 2. Comparative analysis of the effectiveness of regression testing strategies (compiled by the author based on [6, 11, 13, 16]).

Characteristic	Google (TAP + Transition Prediction)	Meta (Conveyor + TestGen-LLM)	Spotify (Backstage + TIA)	Traditional Enterprise (Jenkins + Retest All)
Selection mechanism	ML-based prediction of failure probability	Mutation testing (AI) + canary	Dynamic call graph analysis	Full run / manual selection
Handling of flaky tests	Automatic quarantine and retry	Statistical filtering	Stability monitoring	Manual analysis (high rework)
Feedback latency	Minutes (pre-submit)	Hours (production feedback)	Minutes (PR check)	Hours – days
Scalability	Extreme (monorepository)	Extreme (continuous deployment)	High (microservices)	Low (bottleneck)
Impact on rework	Reduction due to early detection	Reduction due to fast rollback	Reduction due to transparency	Increase due to late detection

Second, a serious constraint is the prevalence of legacy systems with monolithic architectures and a high degree of coupling: under such conditions, even minimal code changes may transitively affect a large portion of the system, which practically nullifies the advantages of Test Impact Analysis, as ensuring an acceptable level of confidence requires retesting up to 80% of the functionality [2, 8].

Additionally, the implementation of intelligent test selection strategies is complicated by the lack of a mature observability infrastructure. Without advanced tracing capabilities, centralized logging, and metrics, it is impossible to construct reliable dependency graphs on which modern TIA algorithms are based; as a result, the system is forced either to overestimate the required testing scope, losing performance gains, or to underestimate risks, reducing reliability.

Finally, security risks remain a significant factor: any automated, probabilistic test selection inherently carries a nonzero risk of defect omission. In domains with heightened reliability requirements — such as financial systems or medical software — this necessitates the introduction of additional layers of control and protective mechanisms: formalization of audit processes, implementation of synthetic monitoring, and other techniques of active observation of system behavior in production, which compensate for the inevitable incompleteness of test coverage.

IV. Conclusion

The analysis of empirical data and case studies of large technology companies confirms the initial hypothesis that, in large-scale software systems, the effectiveness of continuous delivery is not a monotonically linear function of release frequency. As the flow of changes accelerates, the system reaches an inflection point, after which further increasing the deployment frequency without employing adaptive testing strategies does not lead to higher performance but instead increases process entropy: the Change Failure Rate grows, the Rework Rate increases, and the share of useful throughput actually directed at the creation of new value decreases. Thus, delivery speed becomes a factor of degradation if it is not supported by intelligent mechanisms for managing regression risk.

The identified trends indicate a paradigm shift in engineering practice: the industry is gradually abandoning the deterministic principle of testing everything in favor of stochastic risk management focused on verifying those parts of the system that are most likely to have undergone functional or architectural shifts. The practices of Google and Meta demonstrate that for web-scale systems and monorepositories this approach is not optional but effectively the only viable option: attempts to preserve full regression runs on every change lead to the collapse of feedback time and block product evolution. Against this background, the Rework Rate acts not as an auxiliary but as a key integral metric of the health of the development process: a high level of rework combined with high deployment frequency is a direct indicator of a dysfunctional delivery pipeline, signaling hidden costs that are not visible when analyzing only release frequency and time to restore service.

Technological advances in the field of AI radically change the available solution space: the use of large language models for test generation, change analysis, and risk prediction is ceasing to be a competitive advantage and is becoming a technological imperative for teams aspiring to belong to the Elite cluster. Without automated LLM-based support, it becomes practically impossible to simultaneously maintain a high delivery frequency, a low CFR, and a controlled Rework Rate under conditions of increasing code and infrastructure complexity. In parallel, intelligent test selection based on TIA demonstrates a tangible economic effect: a 40–60% reduction in infrastructure resource consumption directly translates into lower operational expenditures, freeing budgets for innovation and increasing the resilience of the engineering ecosystem.

The results obtained lead to specific organizational and technological recommendations for companies aiming to transition into the High/Elite clusters. First, it is necessary to implement automated collection and monitoring of DORA metrics with mandatory inclusion of the Rework Rate among the target KPIs, not as a secondary quality indicator but as one of the key parameters for assessing development processes. Second, targeted investment is required in Platform Engineering class tools (functional analogues of Backstage and TAP) that provide the capability for dynamic Test Impact Analysis at the level of the entire organization rather than individual teams. Third, a policy of zero tolerance for unstable tests must be institutionalized: any flaky tests are subject to immediate automated isolation, labeling, and quarantine with subsequent diagnostics, but not to being ignored or to habituation to their presence. Only a combination of metrics, platform-level solutions, and strict discipline in working with quality makes it possible to achieve a continuous delivery mode

in which a high release frequency does not undermine but instead strengthens the overall resilience of the system.

REFERENCES

1. Shahin, M., Zahedi, M., Babar, M. A., & Zhu, L. (2019). An empirical study of architecting for continuous delivery and deployment. Empirical Software Engineering, 24(3), 1061-1108.
2. Sabau, A. R., Hacks, S., & Steffens, A. (2021). Implementation of a continuous delivery pipeline for enterprise architecture model evolution. Software and Systems Modeling, 20(1), 117-145.
3. 2025 DORA State of AI-assisted Software Development Report. Retrieved from: https://www.honeycomb.io/resources/whitepapers/dora-report-2025 (date accessed: November 5, 2025).
4. Wilkes, B., Milani, A. M. P., & Storey, M. A. (2023, October). A framework for automating the measurement of devops research and assessment (dora) metrics. In 2023 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 62-72). IEEE. https://doi.org/10.1109/ICSME58846.2023.00018.
5. Distilled summary of 2024/2025 Google DORA Report. Retrieved from: https://www.gitclear.com/research/google_dora_2024_summary_ai_impact (date accessed: November 5, 2025).
6. Zolfaghari, B., Parizi, R. M., Srivastava, G., & Hailemariam, Y. (2021). Root causing, detecting, and fixing flaky tests: State of the art and future roadmap. Software: Practice and Experience, 51(5), 851-867. https://doi.org/10.1002/spe.2929.
7. Haas, R., Nömmer, R., Juergens, E., & Apel, S. (2024). Optimization of automated and manual software tests in industrial practice: A survey and historical analysis. IEEE Transactions on Software Engineering, 50(8), 2005-2020. https://doi.org/10.1109/TSE.2024.3418191.
8. Shi, A., Zhao, P., & Marinov, D. (2019, October). Understanding and improving regression test selection in continuous integration. In 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE) (pp. 228-238). IEEE. https://doi.org/10.1109/ISSRE.2019.00031.
9. Accelerate State of DevOps Report 2024. Retrieved from: https://dora.dev/research/2024/ (date accessed: November 8, 2025).
10. DORA Report 2024 – A Look at Throughput and Stability – Alt + E S V - RedMonk, accessed November 29, 2025, https://redmonk.com/rstephens/2024/11/26/dora2024/ (date accessed: November 28, 2025).
11. Speculative Testing at Google with Transition Prediction - Hackthology. Retrieved from: https://hackthology.com/speculative-testing-at-google-with-transition-prediction.html (date accessed: November 28, 2025).
12. Leong, C., Singh, A., Papadakis, M., Le Traon, Y., & Micco, J. (2019, May). Assessing transition-based test selection algorithms at google. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 101-110). IEEE. https://doi.org/10.1109/ICSE-SEIP.2019.00019.
13. Model Freshness Systems at Scale. Retrieved from: https://atscaleconference.com/model-freshness-systems-at-scale/ (date accessed: November 28, 2025).
14. Revolutionizing software testing: Introducing LLM-powered bug catchers. Retrieved from: https://engineering.fb.com/2025/02/05/security/revolutionizing-software-testing-llm-powered-bug-catchers-meta-ach/ (date accessed: November 28, 2025).
15. Verdecchia, R., Lago, P., Ebert, C., & De Vries, C. (2021). Green IT and green software. IEEE software, 38(6), 7-15. https://doi.org/10.1109/MS.2021.3102254.
16. Spotify for Backstage Wrapped 2023: A year of firsts!. Retrieved from: https://backstage.spotify.com/discover/blog/spotify-for-backstage-wrapped-2023/ (date accessed: November 28, 2025).
17. Jürgens, E., Pagano, D., & Göb, A. (2018). Test impact analysis: Detecting errors early despite large, long-running test suites. Whitepaper, CQSE GmbH, 1-6.
18. Muñoz, M., Negrete, M., & Mejía, J. (2019, March). Proposal to avoid issues in the DevOps implementation: A systematic literature review. In World Conference on Information Systems and Technologies (pp. 666-677). Cham: Springer International Publishing.
19. Maclean, L. (2019). Scaling DevOps in Large Enterprises: Challenges and Solutions. International Journal of Artificial Intelligence and Machine Learning, 6(5), 239-248.
20. Azizi, M., & Do, H. (2018, October). ReTEST: A cost effective test case selection technique for modern software development. In 2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE) (pp. 144-154). IEEE.
21. Alshahwan, N., Harman, M., Marginean, A., Tal, R., & Wang, E. (2024, July). Observation-based unit test generation at meta. In Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (pp. 173-184). https://doi.org/10.1145/3663529.3663838.
22. Schwendner, D., Jungwirth, M., Gruber, M., Knoche, M., Merget, D., & Fraser, G. (2025, March). Practical Pipeline-Aware Regression Test Optimization for Continuous Integration. In 2025 IEEE Conference on Software Testing, Verification and Validation (ICST) (pp. 371-381). IEEE. https://doi.org/10.1109/ICST62969.2025.10989002.