A Comparative Study of Regression Testing Techniques: An Industry-Oriented Evaluation of Efficiency, Coverage and Cost

Trina Saha*, Md. Siam

Department of Computer Science and Engineering, R. P. Shaha University, Narayanganj, Dhaka, Bangladesh

Email: trinasahansu@gmail.com*, mdsimonsarkar135@gmail.com

Abstract

Regression testing is an essential component of software maintenance, aimed at ensuring that newly introduced changes do not negatively impact the existing functionality of a system. In today's fast-paced industrial environments, particularly those employing agile methodologies and Continuous Integration/Continuous Deployment (CI/CD) pipelines, the choice of regression testing technique significantly influences project timelines, resource allocation, and product quality. This research investigates and compares five widely adopted regression testing types: Corrective, Retest-All, Selective, Progressive, and Complete Regression Testing. This study's primary objective is to assess each technique's industrial suitability based on key evaluation metrics such as time efficiency, cost of execution, test coverage, automation/tool support, scalability, and risk of fault omission. We adopted a qualitative scoring methodology, grounded in a comprehensive review of several scholarly articles and industry reports. Each testing type was critically analyzed and benchmarked using a comparison matrix to highlight its strengths and limitations. The analysis reveals that while each regression testing method serves distinct use cases, Selective Regression Testing strikes the best balance between efficiency and coverage for modern industrial needs, particularly in projects with frequent releases and constrained testing budgets. The novelty of this study lies in its holistic, literature-backed comparative framework, explicitly tailored to the industry context aspect often overlooked in prior academic evaluations.

Keywords

Regression Testing, Software Maintenance, Industrial Suitability, Selective Regression Testing, Agile CI/CD.

Introduction

Regression testing is a crucial quality assurance practice that ensures new changes such as feature additions, bug fixes, or enhancements do not disrupt existing functionality. It plays a vital role in the maintenance phase of the software development life cycle, preserving software integrity and

Submission:26 November 2025; **Acceptance:** 29 December 2025; **Available online:** December 2025

stability over time (Yoo & Harman, 2012). With the rise of agile methods and CI/CD pipelines, efficient and scalable regression testing has become more critical, yet full test suite execution after every update is costly and time-consuming, especially in large-scale systems. Studies indicate that regression testing can account for up to 50% of overall testing efforts during maintenance, with challenges such as high execution time, increasing costs, limited coverage, and lack of scalability becoming more prominent in larger codebases and shorter release cycles (Do et al., 2010).

In industry, comprehensive regression testing is often unfeasible due to strict timelines, limited resources, and high costs. While full retesting ensures coverage, it is impractical in agile and DevOps settings, creating a challenge to balance cost, coverage, and speed. Prior studies (Rothermel & Harrold, 1997; Gruslu et al., 2017) highlight the need for scalable approaches suited to real-world demands. Over time, methods such as Corrective, Retest-All, Selective, Progressive, and Complete Regression Testing (Saha & Palit, 2019) have emerged, yet comparative analysis of their industrial suitability remains limited.

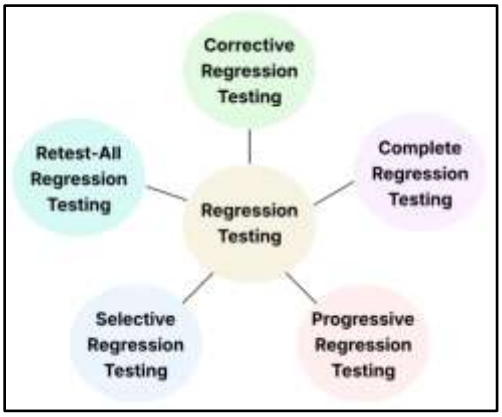

Figure 1. Types of Regression Testing

The choice of regression testing method depends on factors such as project size, development methodology, available resources, and release frequency. Figure 1 shows the types of regression testing. Corrective regression testing re-executes existing test cases when specifications or architecture remain unchanged, offering a cost-effective solution for stable software (Maspupah et al., 2021; Pighin & Marzona, 2005) but may miss hidden defects (Microsoft, 2025). Retest-all regression testing involves re-running the entire test suite, ensuring maximum coverage but at a high cost and time consumption (Rothermel & Harrold, 1996; Lou et al., 2019), typically used before major releases (Gupta et al., 2011). Selective regression testing executes only relevant test cases, reducing execution time without significantly compromising fault detection (Rothermel & Harrold, 1997; Yoo & Harman, 2012), and can save up to 60% of test execution time in agile environments (Poliarush, 2025). Progressive regression testing is applied when both code and tests are modified, ensuring backward compatibility and enhancing defect detection in continuous integration pipelines (Elbaum et al., 2003; Das, 2025). Complete regression testing validates the entire application, ensuring both functional and non-functional aspects are covered. It is resource-intensive but essential in safety-critical domains such as aerospace or healthcare (Leung & White, 1990; IEEE, 2008; Powell & Smalley, 2025).

This study compares five regression testing types based on execution time, cost, resource use, tool support, and agile compatibility. The aim is to identify the most suitable method for modern software development and propose a technique that balances efficiency and effectiveness in practical industrial settings.

Regression testing ensures software quality during frequent updates. Research highlights trade-offs in efficiency, cost, and scalability, with academics focusing on optimization and industry addressing resource, tool, and automation challenges. These findings reveal gaps, stressing the need for industrially relevant comparative analyses. Regression testing has been widely studied due to its critical role in software quality. Prior work has focused on test case selection, prioritization, and minimization to reduce cost and time without sacrificing effectiveness. A seminal contribution by Rothermel & Harrold (1997) introduced a safe regression test selection technique, enabling execution of only affected test cases. Subsequent studies examined challenges and advances: Yoo & Harman (2012) classified regression testing into minimization, selection, and prioritization but noted limited industrial applicability; Do & Rothermel (2006) highlighted cost-benefit trade-offs in prioritization; Elbaum et al. (2002) and Kim & Porter (2002) proposed dynamic and history-based prioritization, respectively. More recent works explored AI and deep learning (Sharif et al., 2021; Tufano et al., 2019) for prioritization, showing potential in CI contexts but limited by data and tooling. Empirical comparisons (Do et al., 2005) and process maturity reviews (Gruslu et al., 2017) further emphasized challenges of adoption. Industrial studies (Garousi et al., 2016a; Feldt et al., 2008; Saha & Palit, 2019) reveal that regression testing is widely practiced but often ad-hoc, constrained by resources, insufficient automation, and skill gaps. Recent research revisited prioritization for CI (Cheng et al., 2024) and mapped performance regression testing challenges (dos Santos et al., 2024).

Despite these advances, few works holistically compare regression testing approaches such as Corrective, Retest-All, Selective, Progressive, and Complete in terms of real-world criteria like cost, time, scalability, and tool support. This study addresses that gap by evaluating these techniques from both literature and industry perspectives.

Despite the variety of regression testing approaches available, there remains a lack of structured comparative analysis focused on their applicability and efficiency in industrial environments. Specifically, little research has investigated how different regression testing types perform in practice under constraints such as cost, execution time, scalability, and support for agile workflows.

This study aims to fill this research gap by:

- Providing a comparative evaluation of five regression testing techniques: Corrective, Retest-All, Selective, Progressive, and Complete.
- Assessing their suitability based on practical industrial criteria such as time efficiency, cost, scalability, and tool support.
- Offering a data-supported recommendation for the most suitable regression testing type for industrial adoption.

By bridging the gap between academic theory and industrial application, this paper intends to guide practitioners in selecting efficient and context-appropriate regression testing strategies.

Methodologies

This study uses a comparative analytical approach to evaluate five regression testing techniques: Corrective, Retest-All, Selective, Progressive, and Complete, focusing on their industrial applicability. Each method is assessed using standardized evaluation criteria derived from literature and industry best practices.

Research Design

To ensure real-world relevance, this study uses qualitative and semi-quantitative criteria from literature and industry to evaluate regression testing. The workflow, from problem identification to conclusion, integrates review, data collection, evaluation, and comparative analysis, ensuring clarity, transparency, and reproducibility. Figure 2 shows the complete workflow.

Figure 2. Overall workflow of the study

As illustrated in Figure 2, the workflow begins with identifying the research gap and establishing clear objectives, followed by an in-depth literature review. The process then moves to the selection of regression testing types and definition of evaluation criteria, which guide data collection and synthesis from both academic and industrial sources. Comparative scoring and analysis are subsequently performed to assess each testing type, culminating in discussions, recommendations, and a conclusion that align with the study's goals.

Evaluation Criteria

The five regression testing types were compared across the following eight industrially relevant dimensions, identified from prior research (Yoo & Harman, 2012; Gruslu et al., 2017; and Do &

Rothermel, 2006): time efficiency, cost of execution, test coverage, tool support and automation, setup complexity, suitability for Agile/CI-CD environments, scalability in large projects, and the risk of missing faults. These dimensions collectively form the comparative analysis framework used in this study, as illustrated in Figure 3.

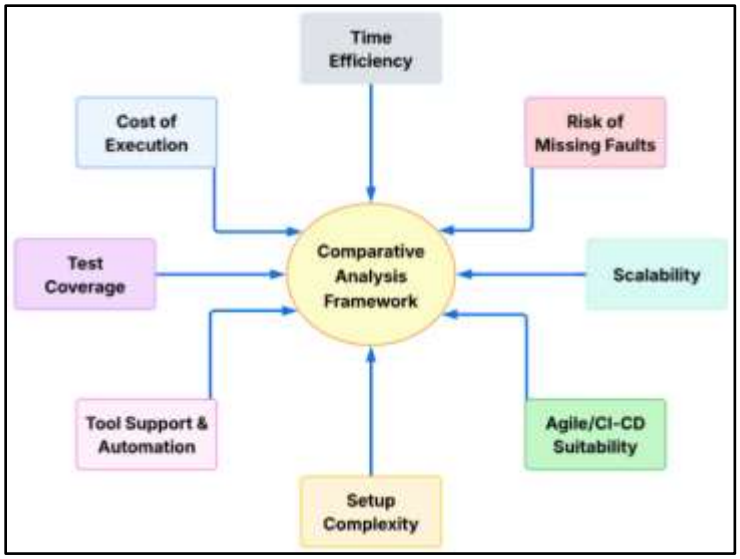

Figure 3. Evaluation Framework

- Time Efficiency - How quickly the testing type can be executed after changes.
- Cost of Execution - Human and computational resource costs involved in running the test suite.
- Test Coverage - The degree to which the testing approach validates modified and unmodified code.
- Tool Support and Automation - Availability of tools and frameworks to automate the testing type.
- Setup Complexity - Technical difficulty in setting up and maintaining the testing process.
- Suitability for Agile/CI-CD - Alignment with modern agile workflows and continuous integration pipelines.
- Scalability in Large Projects - Ability to function efficiently in large, complex codebases.
- Risk of Missing Faults - The likelihood of undetected bugs due to limitations in scope.

Data Collection Approach

The evaluation draws upon:

- Literature Review: A detailed study of several peer-reviewed papers covering regression testing practices, techniques, and their performance in research and industrial environments.
- Expert Knowledge Synthesis: Insights synthesized from published industrial case study (Gruslu et al., 2017) and tool documentation (e.g., Selenium, JUnit, Jenkins).
- Analytical Scoring: Each testing type was qualitatively rated (e.g., "High", "Medium", "Low") across the eight criteria, informed by referenced empirical studies and expert discussion.

Results and Discussion

This section reviews five regression testing approaches: Corrective, Retest All, Selective, Progressive, and Complete. They are evaluated on factors such as cost, execution time, coverage, tool support, scalability, and fault detection. Levels are used instead of percentages since test case sizes are not specified.

Table 1. Comparative Analysis of Regression Testing Types with Referenced Justifications

Table 1	Criteria	Corrective	Retest-All	Selective	Progressive	Complete	Supporting References
1	Time Efficiency	High	Very Low	High	Medium	Low	(Yoo & Harman, 2012; Rothermel & Harrold, 1997; Garousi et al., 2016b; Reinisch et al., 2008)
2	Cost of Execution	Low	High	Medium	Medium	Very High	(Yoo & Harman, 2012; Do & Rothermel, 2006; Elbaum et al., 2002; Garousi et al., 2016b; Reinisch et al., 2008)
3	Test Coverage	Medium	Very High	Medium	High	Very High	(Yoo & Harman, 2012; Rothermel & Harrold, 1997; Do & Rothermel, 2006; Kruchten et al., 2002)
4	Tool Support & Automation	Moderate	High	High	High	Moderate	(Yoo & Harman, 2012; Garousi et al., 2016b; Reinisch et al., 2008; Garousi et al., 2018)
5	Complexity of Setup	Low	Low	Medium	High	Very High	(Elbaum et al., 2002; Garousi et al., 2016b; Reinisch et al., 2008; Ren et al., 2019)
6	Suitability for Agile/CI-CD	Moderate	Poor	Excellent	Excellent	Poor	(Yoo & Harman, 2012; Garousi et al., 2016b; Elbaum et al., 2002; Garousi et al., 2018)
7	Scalability in Large Projects	Medium	Low	High	High	Medium	Elbaum et al., 2002; Garousi et al., 2016b; Reinisch et al., 2008; Ren et al., 2019)
8	Risk of Missing Faults	Medium	Low	Medium	Low	Very Low	(Yoo & Harman, 2012; Rothermel & Harrold, 1997; Do & Rothermel, 2006; Kruchten et al., 2002)

Table 1 presents a comparative analysis of regression testing types based on efficiency, cost, coverage, scalability, and Agile/CI/CD suitability. Retest-All and Complete strategies provide very high coverage but incur significant time and cost, making them less practical for fast development. Selective and Progressive approaches balance efficiency, automation, and adaptability, fitting modern CI/CD pipelines. Corrective testing is the most cost-effective but limited in scalability and fault detection. Overall, no single strategy is universally optimal; selection should consider project scale, development context, and organizational priorities.

Graphical Comparison

A comparative analysis was conducted to evaluate the strengths and weaknesses of different regression testing approaches. The assessment considered execution time, cost, test coverage, tool support and automation, setup complexity, Agile/CI-CD suitability, scalability for large projects, and fault detection risk. Each testing type, including Corrective, Retest-All, Selective, Progressive, and Complete, was rated on a five-point scale from 1 (Very Low/Poor) to 5 (Very High/Excellent) based on literature and expert opinion.

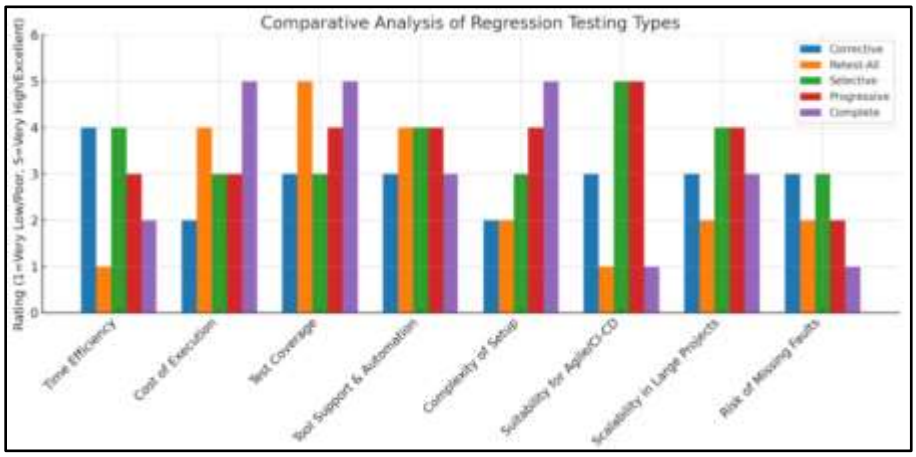

Figure 4. Comparison Analysis of Regression Testing Types

As shown in Figure 4, no single regression testing type excels across all criteria. Retest-All and Complete approaches offer the highest test coverage but have low efficiency and high cost, limiting their use in large-scale or Agile environments. Selective and Progressive methods provide better scalability and efficiency, with slightly lower coverage. Corrective testing delivers balanced performance but may not suit environments requiring rapid iteration. These results indicate that selecting a regression testing strategy should be context-driven, balancing coverage, cost, and adaptability to project needs.

Reference Mapping

To ensure credibility and industrial relevance, this study synthesizes insights from peer-reviewed research and industry case studies. Table 2 maps the cited references to their contributions in regression testing, tracing the origin of concepts, methodologies, and findings. This mapping supports transparency, reproducibility, and ensures that the comparative analysis is grounded in both established literature and validated industrial practices.

Table 2. Reference Mapping

Ref.	Authors	Contributions
1	(Yoo & Harman, 2012)	Comparative survey of regression testing methods in industry
2	(Rothermel & Harrold, 1997)	Test coverage and safe regression testing selection
3	(Do & Rothermel, 2006)	Cost analysis and prioritization strategies
4	(Garousi et al., 2016b)	Industrial adoption, scalability, and limitations
5	(Garousi et al., 2018)	Automation and CI-based prioritization
6	(Elbaum et al., 2002)	Empirical prioritization vs. setup complexity
7	(Reinisch et al., 2008)	Execution time and automation challenges

8	(Ren et al., 2019)	Data-driven test case selection
9	(Kruchten et al., 2002)	Impact analysis and fault detection coverage

Survey Findings and Recommendation

The comparative analysis shows that each regression testing type has distinct advantages and limitations depending on the industrial context. Corrective Regression Testing is simple and cost-effective, suitable for small to medium projects with infrequent bug fixes. Retest-All Regression Testing provides high reliability but incurs significant time and cost, limiting its use in agile or continuous delivery environments. Selective Regression Testing balances efficiency and coverage, fitting well with DevOps and CI/CD pipelines. Progressive Regression Testing is effective for projects with evolving requirements and iterative development. Complete Regression Testing delivers maximum coverage but is feasible only in high-budget, high-criticality systems such as aerospace or medical software. Overall, cost efficiency and time-to-market are key factors, with agile organizations favoring Selective and Progressive approaches, while safety-critical domains rely on Complete Regression Testing despite its high resource demands.

Based on the analysis, **Selective Regression Testing** is recommended as the most industrially suitable approach for the majority of modern software development environments. This recommendation is grounded in its ability to:

- Reduce execution time by focusing only on impacted modules.
- Maintain a high level of fault detection accuracy.
- Support automation tools and CI/CD workflows effectively.
- Offer a cost-performance balance suitable for both SMEs and large enterprises.

For safety-critical systems (e.g., aerospace, healthcare, autonomous driving), Complete Regression Testing remains the most reliable option, whereas Progressive Regression Testing is highly beneficial in rapidly evolving product environments.

Limitations and Future Scope

This semi-empirical study is based on literature and industrial reports rather than experiments. Findings may vary with project-specific factors such as domain, budget, and tool availability. No primary data were collected, and the evaluation assumes access to modern automation frameworks. Future research should emphasize large-scale industrial validation and the development of hybrid or adaptive regression testing strategies. Promising directions include AI-driven method selection, emerging architectures such as microservices and DevOps pipelines, and detailed cost-benefit analysis. Improving tool interoperability and integration is also crucial for scalability and practical adoption in evolving software ecosystems.

Conclusion

Regression testing ensures that system changes do not introduce faults. This study compared five strategies, including Corrective, Retest-All, Selective, Progressive, and Complete, using academic

insights and industry data. The analysis shows that Selective Regression Testing offers the best balance of coverage, efficiency, and cost for agile and CI/CD environments. Progressive Regression Testing is valuable for projects with evolving requirements, while Complete Regression Testing is essential in safety-critical domains where reliability outweighs cost and time. No single strategy is universally optimal; suitability depends on project scale, domain criticality, budget, and development methodology. Organizations should adopt context-driven approaches, and future research should explore hybrid or adaptive models leveraging automation, AI-driven test selection, and predictive analytics to improve efficiency and fault detection.

Acknowledgements

There is no grant or funding bodies to be acknowledged for preparing this paper. The authors acknowledge with gratitude the contributions of the researchers and authors whose published works were reviewed and analyzed in this study.

References

Cheng, R., Wang, S., Jabbarvand, R., & Marinov, D. (2024). Revisiting test-case prioritization on long-running test suites. *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)* (pp. 1–12). ACM. https://doi.org/10.1145/3650212.3680307

Das, S. (2025, January). CI/CD with Jenkins. *Jenkins Blog.* https://www.jenkins.io/blog/

Do, H., & Rothermel, G. (2006). On the use of mutation faults in empirical assessments of test case prioritization techniques. *IEEE Transactions on Software Engineering, 32*(9), 733–752. https://doi.org/10.1109/TSE.2006.92

Do, H., Elbaum, S., & Rothermel, G. (2005). Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. *Empirical Software Engineering, 10*(4), 405–435. https://doi.org/10.1007/s10664-005-3861-2

Do, H., Mirarab, S., Tahvildari, L., & Rothermel, G. (2010). The effects of time constraints on test case prioritization: A series of controlled experiments. *IEEE Transactions on Software Engineering, 36*(5), 593–617. https://doi.org/10.1109/TSE.2010.58

dos Santos, L. B. R., Felício, T. G., de Andrade, R. M. C., de Souza, É. F., de Mello, R. M., & Figueiredo, E. (2024). Performance regression testing initiatives: A systematic mapping study. *Information and Software Technology, 170,* 107233. https://doi.org/10.1016/j.infsof.2024.107641

Elbaum, S., Karre, S., & Rothermel, G. (2003). Improving web application testing with user session data. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)* (pp. 49–60). ACM. https://doi.org/10.1109/ICSE.2003.1201188

Elbaum, S., Malishevsky, A., & Rothermel, G. (2002). Test case prioritization: A family of empirical studies. *IEEE Transactions on Software Engineering, 28*(2), 159–182. https://doi.org/10.1109/32.988497

Feldt, R., Torkar, R., Angelis, L., & Samuelsson, M. (2008). Towards individualized software engineering: Empirical studies should collect psychometrics. In *Proceedings of the*

International Conference on Software Engineering (ICSE) (pp. 281–290). IEEE. https://doi.org/10.1145/1370114.1370127

Garousi, V., Felderer, M., & Hacaloğlu, T. (2016a). Software test maturity assessment: Literature review and industrial evaluation. *Journal of Systems and Software, 117,* 331–351. https://doi.org/10.1016/j.infsof.2017.01.001

Garousi, V., Özkan, R., & Betin-Can, A. (2018). Multi-objective regression test selection in practice: An empirical study in the defense software industry. *Information and Software Technology, 103,* 40–54. https://doi.org/10.1016/j.infsof.2018.06.007

Garousi, V., Petersen, K., & Ozkan, B. (2016b). Challenges and best practices in industry–academia collaborations in software engineering: A systematic literature review. *Information and Software Technology, 79,* 106–127. https://doi.org/10.1016/j.infsof.2016.07.006

Gruslu, V., Felderer, M., & Hacaloglu, T. (2017). Software test maturity assessment and test process improvement: A multivocal literature review. *Information and Software Technology, 85,* 16–33. https://doi.org/10.1016/j.infsof.2017.01.001

Gupta, P., Ivey, M., & Penix, J. (2011, June). Testing at the speed and scale of Google. *Google Testing Blog.* https://testing.googleblog.com/2011/06/testing-at-speed-and-scale-of-google.html

IEEE. (2008). *IEEE standard for software and system test documentation (IEEE Std 829-2008)* (pp. 1–150). IEEE. https://doi.org/10.1109/IEEESTD.2008.4578383

Kim, J., & Porter, A. (2002). A history-based test prioritization technique for regression testing in resource-constrained environments. In *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA)* (pp. 119–129). ACM. https://doi.org/10.1145/581339.581357

Kruchten, P., Hilliard, R., Kazman, R., Kozaczynski, W., Obbink, H., & Ran, A. (2002). Workshop on methods and techniques for software architecture review and assessment (SARA). In *Proceedings of the 24th International Conference on Software Engineering (ICSE '02)* (p. 675). ACM. https://doi.org/10.1145/581339.581439

Leung, H. K. N., & White, L. J. (1990). A study of integration testing and software regression at the integration level. In *Proceedings of the Conference on Software Maintenance* (pp. 290–301). IEEE. https://doi.org/10.1109/ICSM.1990.131377

Lou, Y., Chen, J., Zhang, L., & Hao, D. (2019). A survey on regression test-case prioritization. *Advances in Computers, 113,* 1–46. https://doi.org/10.1016/bs.adcom.2018.10.001

Maspupah, A., Rahmani, A., & Min, J. (2021). Comparative study of regression testing tools feature on unit testing. *Journal of Physics: Conference Series, 1869*(1), 012098. https://doi.org/10.1088/1742-6596/1869/1/012098

Microsoft. (2025). *Azure DevOps Server.* https://learn.microsoft.com/azure/devops/server/

Pighin, M., & Marzona, A. (2005). Optimizing test to reduce maintenance. In *Proceedings of the IEEE International Conference on Software Maintenance (ICSM)* (pp. 465–472). IEEE. https://doi.org/10.1109/ICSM.2005.69

Poliarush, M. (2025, August). Agile regression testing: Process & best practices. *Software Testing Help.* https://testomat.io/blog/agile-regression-testing/

Powell, P., & Smalley, I. (2025, June). What is regression testing? *TestingXperts Blog.* https://www.testingxperts.com/blog/what-is-regression-testing/

Reinisch, C., Granzer, W., Praus, F., & Kastner, W. (2008). Integration of heterogeneous building automation systems using ontologies. In *Proceedings of the 34th Annual Conference of IEEE*

Industrial Electronics (pp. 2736–2741). IEEE. https://doi.org/10.1109/IECON.2008.4758391

Ren, J., Xu, Z., Yu, C., Lin, C., Wu, G., & Tan, G. (2019). Execution allowance-based fixed-priority scheduling for probabilistic real-time systems. *Journal of Systems and Software, 152,* 120–133. https://doi.org/10.1016/j.jss.2019.03.001

Rothermel, G., & Harrold, M. J. (1996). Analyzing regression test selection techniques. *CORE Repository, University of Nebraska–Lincoln.* https://doi.org/10.1109/32.536955

Rothermel, G., & Harrold, M. J. (1997). A safe, efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology, 6*(2), 173–210. https://doi.org/10.1145/248233.248262

Saha, T., & Palit, R. (2019). Practices of software testing techniques and tools in Bangladesh software industry. In *Proceedings of the IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (pp. 1–10). IEEE. https://doi.org/10.1109/CSDE48274.2019.9162355

Sharif, A., Marijan, D., & Liaaen, M. (2021). DeepOrder: Deep learning for test case prioritization in continuous integration testing. *arXiv Preprint, arXiv:2110.07443.* https://doi.org/10.48550/arXiv.2110.07443

Tufano, M., Watson, C., Bavota, G., Di Penta, M., White, M., & Poshyvanyk, D. (2019). An empirical study on learning bug-fixing patches in the wild via neural machine translation. In *Proceedings of the International Conference on Software Engineering (ICSE)* (pp. 1–11). IEEE. https://doi.org/10.48550/arXiv.1812.08693

Yoo, S., & Harman, M. (2012). Regression testing minimization, selection and prioritization: A survey. *Software Testing, Verification and Reliability, 22*(2), 67–120. https://doi.org/10.1002/stvr.430