International Journal Software Engineering and Computer Science (IJSECS)
6 (1), 2026, 140-149
Published Online April 2026 in IJSECS (http://www.journal.lembagakita.org/index.php/ijsecs)
P-ISSN: 2776-4869, E-ISSN: 2776-3242. DOI: https://doi.org/10.35870/ijsecs.v6i1.6810.

RESEARCH ARTICLE Open Access

Automated Regression Testing Procedure Based on ISO/IEC 29119 to Improve Software Testing Efficiency in a Software Development Environment

Galuh Oka Safitri [1*], **Leni Susanti** [2]

[1*,2] Information Systems Study Program, Faculty of Computer Science, Universitas Pamulang, South Tangerang City, Banten Province, Indonesia.

**Corresponding author*: dosen02617@unpam.ac.id.

Received: March 14, 2026; Accepted: April 1, 2026; Published: April 10, 2026.

Abstract: The growing complexity of modern software systems has increased the importance of effective software testing practices to ensure system reliability and quality throughout the development lifecycle. Among various testing activities, regression testing plays a crucial role in verifying that newly introduced changes do not negatively affect previously functioning system components (Yoo & Harman, 2012). However, in many organizations regression testing is still conducted manually, which can be time-consuming and may delay software release cycles. This study aims to design an automated regression testing procedure based on the ISO/IEC 29119 software testing standard in order to improve testing efficiency within a software development environment. A case study was conducted at PT. ABCD to examine the existing regression testing practices. Data were collected through semi-structured interviews with members of the Quality Assurance (QA) team, direct observation of testing activities, and analysis of testing documentation. The research process consisted of analyzing the current regression testing workflow, performing gap analysis with the ISO/IEC 29119 framework, and designing an automated regression testing procedure aligned with the standard. The study focused on five system modules comprising 420 regression test cases. The results indicate that manual regression testing required approximately eight working days to complete, with an average execution time of about nine minutes per test case. After implementing the proposed automated testing procedure, the execution time was reduced to approximately one day, or about 1.14 minutes per test case, resulting in an efficiency improvement of approximately 87.5%. These findings suggest that integrating automated regression testing with a structured testing framework such as ISO/IEC 29119 can significantly improve testing efficiency while also supporting better documentation, traceability, and process consistency. From a practical perspective, the proposed approach may help development teams accelerate testing cycles and support faster delivery of software updates in dynamic development environments.

Keywords: Regression Testing; Test Automation; ISO/IEC 29119; Software Testing Process; Software Testing Efficiency.

1. Introduction

The rapid development of software systems has significantly transformed how organizations design, develop, and maintain applications. Modern software development environments demand faster delivery cycles while maintaining high levels of reliability and quality. As a result, testing activities must evolve to keep pace with frequent system updates and continuous improvements in software functionality. Among various testing activities, regression testing plays a critical role in ensuring that modifications to the software do not negatively affect existing functionalities (Humble & Farley, 2010; Shahin *et al.*, 2017). Regression testing is typically performed after software updates, bug fixes, or the addition of new features. Its primary objective is to verify that previously tested components continue to function correctly after changes have been introduced to the system. However, regression testing can become increasingly complex as software systems grow larger and more interconnected. Executing a large number of test cases repeatedly may require significant time and effort, particularly when the testing process is conducted manually (Gami *et al.*, 2025; Jain, 2023). This situation often creates challenges for software development teams that must balance rapid development cycles with the need to maintain software quality.

To address these challenges, many organizations have started adopting automated testing approaches. Test automation can reduce the time required to execute repetitive test cases and improve the efficiency of the overall testing process (Ammann & Offutt, 2017; Myers, 2006). In many organizations, however, regression testing is still performed manually. Manual testing requires testers to repeatedly execute a large number of test cases whenever system changes occur — an approach that can be time-consuming and inefficient, particularly in development environments that implement CI/CD practices where code changes occur frequently (Yoo & Harman, 2012). Automated testing enables test cases to be executed with minimal human intervention, which can substantially reduce testing time while improving the consistency and repeatability of results (Parekh & Saxena, 2024). Recent studies have proposed various techniques to further improve regression testing efficiency, including regression test selection, test case prioritization, and the application of machine learning techniques to predict fault-prone areas in software systems (Lima & Vergilio, 2020; Pan *et al.*, 2022).

Despite these advancements, the implementation of automated regression testing in practice is often focused on technical tools or optimization techniques rather than on structured testing procedures aligned with established testing standards. In particular, limited attention has been given to how automated regression testing processes can be designed in accordance with recognized software testing frameworks such as the ISO/IEC/IEEE 29119 standard. This standard provides comprehensive guidelines for software testing processes, including test planning, test design, test execution, and test reporting (ISO/IEC/IEEE 29119-1, 2013; ISO/IEC/IEEE 29119-2, 2013).

This study designed an automated regression testing procedure based on the ISO/IEC 29119 software testing framework with the aim of improving regression testing efficiency within a software development environment. A case study approach was adopted at PT. ABCD to analyze existing regression testing practices and identify gaps between current practices and the structured testing processes recommended by the standard. Based on that analysis, a structured automated regression testing procedure was proposed to support a more systematic testing workflow while reducing the time required to execute regression test cases. By pairing automated testing practices with a standardized testing framework, the study shows how regression testing activities can be performed more efficiently while maintaining proper documentation, traceability, and consistency throughout the testing process.

2. Literature Review

Several studies have explored different approaches to improving regression testing efficiency in modern software development environments. Regression test selection is one of the most widely studied approaches. This technique aims to identify a subset of test cases that need to be executed after system modifications. Previous studies have shown that machine learning techniques can effectively predict which test cases are more likely to detect faults, thereby reducing the number of tests executed during regression testing cycles (Pan *et al.*, 2022; Zhang *et al.*, 2022). Another widely used approach is test case prioritization. This method focuses on determining the order in which test cases should be executed so that test cases with a higher probability of detecting faults are executed earlier. Research has demonstrated that test case prioritization can significantly reduce the time required to identify defects in regression testing processes (Lima & Vergilio, 2020).

Automation frameworks have also been widely adopted to support regression testing activities. Automated testing frameworks allow test cases to be executed automatically whenever code changes occur, enabling development teams to receive rapid feedback regarding system quality during the development

process (Gundecha, 2015). The integration of testing activities within CI/CD pipelines has been widely discussed in the literature as well. Continuous testing approaches allow automated regression tests to be executed during each build process, thereby improving software reliability and reducing the risk of deployment failures (Shahin *et al.*, 2017). In addition, recent studies have explored the use of machine learning techniques to optimize regression testing processes. Machine learning models can assist in identifying fault-prone components, predicting software defects, and prioritizing test cases within automated testing pipelines (Panichella *et al.*, 2017). Recent studies have further investigated the role of machine learning and data-driven techniques in improving regression testing efficiency. Hasnain *et al.* (2021) conducted a systematic literature review on functional requirement-based test case prioritization and found that prioritization techniques can significantly reduce regression testing effort while maintaining effective fault detection capability. Alshammari and Harman (2023) examined the application of data-driven regression testing strategies in modern software development environments, reporting that such approaches can improve testing efficiency and support faster feedback cycles within continuous integration workflows. These studies point toward the growing importance of intelligent and automated approaches for managing large regression test suites in evolving software systems. Although previous studies demonstrate that techniques such as regression test selection, test case prioritization, and automation frameworks can improve testing efficiency, most of these approaches primarily focus on technical optimization or algorithmic improvements. Regression test selection techniques aim to reduce the number of test cases that must be executed after system modifications while maintaining fault detection capability (Yoo & Harman, 2012). Similarly, test case prioritization approaches attempt to schedule test cases in an order that increases the likelihood of detecting defects earlier in the testing process (Lima & Vergilio, 2020; Pan *et al.*, 2022).

Despite these advancements, relatively fewer studies examine how automated regression testing practices can be implemented within a structured testing framework that ensures systematic testing activities and proper documentation. In contrast to prior studies that mainly emphasize technical or tool-based solutions, this research focuses on the design of an automated regression testing procedure aligned with the ISO/IEC 29119 software testing standard. By pairing automated testing practices with a standardized testing framework, this study aims to provide a more structured approach to regression testing while improving testing efficiency and supporting better traceability and documentation of testing activities.

3. Methodology

3.1 Research Design

This research adopts a case study approach to examine the regression testing process carried out within a software development organization. The case study method was selected because it allows testing practices to be analyzed within their actual working environment — capturing not only how regression testing is formally described, but how it is actually performed, including the operational challenges that rarely appear in official documentation. A purely experimental or survey-based design would have missed that texture entirely. The primary objective was to design an automated regression testing procedure aligned with the ISO/IEC 29119 software testing standard. The research followed a descriptive and analytical path: existing practices were first documented in sufficient detail to establish how testing currently operated, then examined against the standard's recommendations to identify where the process fell short. The findings from that analysis served as the foundation for designing an automated regression testing procedure suited to the organization's context.

3.2 Data Collection

Three data collection techniques were applied to build a comprehensive picture of the existing regression testing practices. Semi-structured interviews were conducted with five participants directly involved in regression testing — three QA engineers and two software developers. The interviews examined the current testing workflow, the tools in use, and the practical difficulties encountered when regression testing is performed manually, with questions focused on testing procedures, test case execution practices, and documentation management. These were complemented by direct observation of regression testing activities, through which the researcher examined how test cases were executed, how results were recorded, and how testers and developers interacted during testing cycles. Observed behavior sometimes diverged from what participants described in interviews, and those discrepancies proved informative. Document analysis was also conducted on testing artifacts including test case documentation, testing reports, system requirement documents, and change request records, alongside a targeted literature review focused on the ISO/IEC 29119 standard and prior studies on automated regression testing.

3.3 Research Procedure

The research proceeded through four sequential stages. The existing regression testing process was mapped in sufficient detail to document the workflow, the roles involved, the sequence of tasks, and the tools currently in use. That mapping then served as the basis for a gap analysis, in which current practices were compared against the processes recommended in ISO/IEC 29119 to identify where the organization's approach diverged from the standard's expectations. Based on those findings, an automated regression testing procedure was designed — covering regression test planning, development of automated test scripts, execution of automated tests, and documentation of testing results. The design was deliberately constrained by the organization's existing development practices; a procedure that cannot be realistically adopted offers little value regardless of how well it performs on paper. The procedure was then validated through discussions with the QA team to assess its feasibility and compatibility with the existing workflow. Feedback from those discussions was used to refine the procedure before the efficiency evaluation was carried out. The five modules selected for analysis — chosen based on their frequency of modification, volume of associated test cases, and direct impact on core business processes — collectively contained 420 regression test cases, which served as the basis for comparing manual and automated regression testing performance.

3.4 Research Framework

The stages described above form a structured research sequence that begins with observation and problem identification, moves through literature study, data collection, and gap analysis, and arrives at the design of the proposed automated regression testing procedure. The overall process is illustrated in Figure 1.

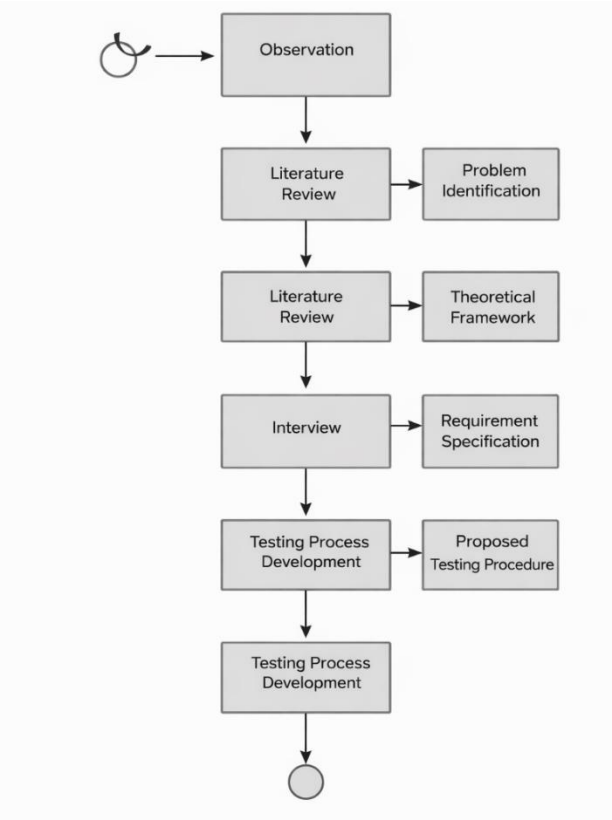

Figure 1. Research Framework

As shown in Figure 1, each stage builds directly on the findings of the previous one — a deliberate design choice that ensured the proposed procedure remained grounded in the organization's actual testing conditions rather than constructed in abstraction from them.

4. Result and Discussion

4.1 Results

4.1.1 Overview of the Project and Scope of Regression Testing

The study was conducted within an internal system development project at PT. ABCD, where feature changes occur frequently across development iterations — ranging from minor adjustments to substantial

modifications of core modules. Under those conditions, regression testing is not a peripheral activity; it is the primary mechanism for verifying that new changes have not destabilized previously functioning components. The research focused on five modules identified by the QA team as the most frequently modified and most consequential for organizational operations: CIF Individual, New Customer Registration, Mobile Banking Activation, Mobile Banking Registration, and Print Report. These modules were selected based on discussions with the QA team and a review of system change records from the preceding six months, and they consistently appeared as the primary objects of regression testing in every development iteration. The distribution of test cases across modules is presented in Table 1.

Table 1. Modules and Number of Test Cases

Module	Test Cases
CIF Individual	120
New Customer Registration	95
Mobile Banking Activation	80
Mobile Banking Registration	75
Print Report	50
Total	420

These 420 test cases formed the basis for comparing manual and automated regression testing execution times.

4.1.2 Analysis of the Existing Regression Testing Process

Regression testing at PT. ABCD was conducted entirely by hand at the time of this study. The process was triggered by system modifications and aimed to confirm that previously stable modules continued to function correctly after changes were introduced. The organization operated at CMMI Level 3 — a maturity level that implies a degree of process discipline — yet no formal procedure existed that defined regression testing as a structured, automation-supported activity. Test case selection relied heavily on individual tester judgment rather than any documented selection strategy. That gap between stated maturity and actual practice is worth noting; process certification does not automatically translate into structured testing behavior. The existing workflow, illustrated in Figure 2, began with a feature change request, followed by preparation of a Business Requirement Document (BRD). After requirements validation, development tasks were assigned and executed. Once development was complete, the testing team performed regression tests manually. Successful results led to tester verification and user acceptance testing (UAT); failed results returned the task to developers for correction.

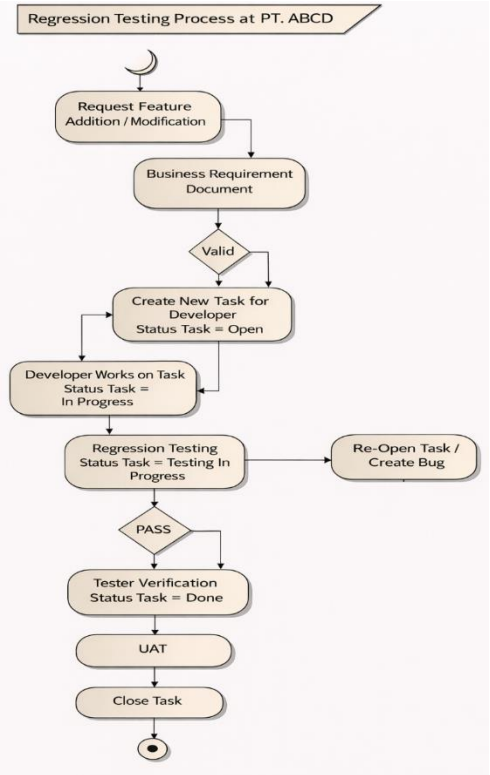

Figure 2. Existing Regression Testing Process at PT. ABCD

Several structural limitations characterized this process: test case selection was based primarily on tester intuition rather than a defined strategy; test cases were executed sequentially with no parallelization; testing results were documented manually, creating traceability gaps; and no integration existed between the testing process and version control systems. Taken together, these limitations meant that regression testing consumed a disproportionate share of the development cycle — and the problem worsened as the system grew.

4.1.3 Gap Analysis of Existing Regression Testing Process with ISO/IEC 29119

A gap analysis was conducted by comparing the existing testing practices at PT. ABCD against the processes specified in ISO/IEC/IEEE 29119, drawing on interview data, direct observation, and document analysis. The results were unambiguous in several areas. Regression testing was performed without a formally documented testing plan. Test case selection followed no defined strategy. Monitoring of test execution was informal, and the documentation of results — while present — lacked the structure needed for meaningful traceability across development cycles. These are not minor procedural gaps; they represent the difference between a testing process that can be audited, improved, and scaled, and one that depends entirely on the knowledge and habits of individual testers. Table 2 summarizes the comparison between current practices and ISO/IEC 29119 recommendations across four testing activities.

Table 2. Gap Analysis of Existing Regression Testing Process Based on ISO/IEC 29119

Testing Activity	Existing Practice	ISO/IEC 29119 Recommendation	Identified Gap
Test Planning	No formal regression testing plan	Structured test planning	Lack of formal planning
Test Monitoring	Informal monitoring	Defined monitoring activities	Limited monitoring mechanism
Test Implementation	Manual execution of test cases	Structured test implementation	Automation not implemented
Test Reporting	Manual documentation	Standardized reporting	Limited documentation traceability

Without a formal plan, testing scope is undefined and difficult to communicate across teams. Without structured monitoring, execution status remains opaque to project stakeholders. Without standardized reporting, results cannot be reliably compared across iterations or provide the audit trail that quality-conscious organizations require. The proposed procedure addresses each of these gaps directly.

4.1.4 Analysis of Manual Regression Testing Execution Time

Execution time data for manual regression testing were collected through direct observation of QA activities. The results, broken down by module, are presented in Table 3.

Table 3. Manual Regression Testing Execution Time

Module	Test Cases	Execution Time
CIF Individual	120	2 Days
New Customer Registration	95	2 Days
Mobile Banking Activation	80	1.5 Days
Mobile Banking Registration	75	1.5 Days
Print Report	50	1 Day
Total	420	8 Days

The average execution time per test case was calculated as follows:

$$\text{Average time per test case} = \frac{8 \text{ days}}{420} = 0.019 \text{ days per test case}$$

Converting to working hours (1 day = 8 hours):

$$0.019 \times 8 = 0.152 \text{ hours} \approx 9 \text{ minutes per test case}$$

Nine minutes per test case, across 420 cases, executed after every significant code change. In a development environment where modifications occur across every iteration, that figure is not sustainable.

4.1.5 Implementation of Automated Regression Testing

The proposed automated regression testing procedure was designed by analyzing the testing processes defined in ISO/IEC 29119 — including expected outcomes, required activities, and documentation artifacts — and adapting those processes to the specific context of PT. ABCD's development environment. The regression testing process as specified by the ISO/IEC 29119 framework is illustrated in Figure 3, while the adapted procedure designed for PT. ABCD is presented in Figure 4.

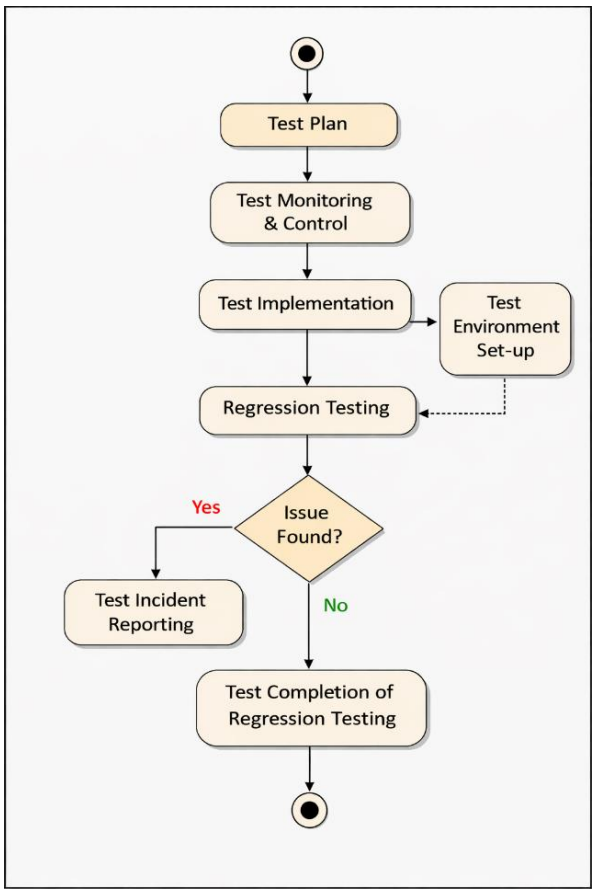

Figure 3. Regression Testing Process Based on ISO/IEC 29119

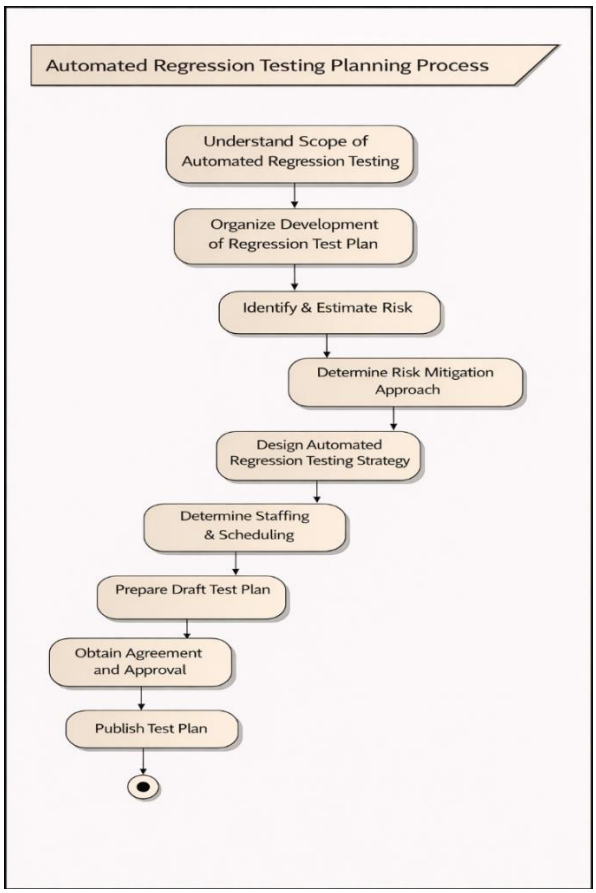

Figure 4. Proposed Automated Regression Testing Planning Process at PT. ABCD

All 420 test cases were converted into automated scripts using Katalon Studio, with script management handled through GitKraken as the version control system — directly addressing one of the more significant gaps identified in the earlier analysis, namely the absence of any integration between testing activities and version control. Execution time after automation was measured and is presented in Table 4.

Table 4. Automated Regression Testing Execution Time

Module	Scripts	Execution Time
CIF Individual	120	0.25 Days
New Customer Registration	95	0.25 Days
Mobile Banking Activation	80	0.20 Days
Mobile Banking Registration	75	0.20 Days
Print Report	50	0.10 Days
Total	420	1 Day

The average execution time per automated test case:

$$\frac{1 \text{ day}}{420} = 0.00238 \text{ days per test case}$$

Converted to working hours:

$$0.00238 \times 8 = 0.019 \text{ hours} \approx 1.14 \text{ minutes per test case}$$

The efficiency gain:

$$\text{Efficiency} = \frac{8-1}{8} \times 100\% = 87.5\%$$

Eight days reduced to one. That is not a marginal improvement — it is a structural change in what the testing team can accomplish within a single development iteration.

4.2 Discussion

The 87.5% reduction in execution time is the most visible result of this study, but it is probably not the most consequential one. Speed gains from automation are well-established; Fewster and Graham (1999) documented them, and subsequent studies have confirmed the pattern repeatedly. What the present study adds is evidence that those gains hold when automation is embedded within a structured, standards-aligned procedure — and that the procedural dimension produces benefits beyond speed alone. Yoo and Harman (2012) demonstrated that techniques such as regression test selection and test case prioritization can improve testing effectiveness by reducing unnecessary execution while maintaining fault detection capability, yet neither study gave sustained attention to how automated testing practices should be governed within a formal process structure. That is precisely the gap this study addresses. Many existing studies focus on technical or algorithmic aspects of regression testing optimization — prioritization algorithms, selection heuristics, automated script generation. These contributions are real, but they tend to treat the testing process as a technical problem rather than an organizational one. In practice, development teams require not only fast test execution but also well-documented, auditable procedures that can be communicated, reviewed, and improved over time. A suite of automated scripts without a governing process is faster than manual testing, but it remains fragile — dependent on whoever wrote the scripts and opaque to anyone who did not.

The procedure proposed here addresses that fragility by aligning automated testing practices with the ISO/IEC 29119 framework, which specifies not just what should be tested but how testing activities should be planned, monitored, and recorded. The result is a testing process that is both faster and more transparent — one where test scope is formally defined, execution is monitored against a plan, and results are documented in a standardized format that supports traceability across development cycles. Alshammari and Harman (2023) and Hasnain *et al.* (2021) both pointed toward the value of structured, data-driven approaches to regression testing management; the present study operationalizes that argument within a specific organizational context. Several limitations should be stated plainly. The research was conducted within a single organization, and the generalizability of the findings to other development environments remains uncertain. The evaluation measured execution time as the primary indicator of efficiency; defect detection effectiveness and the long-term maintenance costs of automated test scripts were not analyzed. Maintaining 420 automated scripts across a codebase that changes frequently is non-trivial work, and that cost should not be invisible in any honest assessment of automation's benefits. Future research could apply the proposed procedure across multiple organizational contexts and attend more carefully to those maintenance dimensions — as well as to the integration of automated regression testing within CI/CD pipelines, where tests run automatically on every build.

5. Conclusion and Recommendations

This study designed an automated regression testing procedure grounded in the ISO/IEC 29119 software testing standard and evaluated its effect on testing efficiency at PT. ABCD. The work proceeded through three stages: analysis of the existing regression testing process, gap analysis against the ISO/IEC 29119 framework, and design of a structured automated procedure that addressed the identified gaps. The results were clear. Manual regression testing across 420 test cases in five modules required eight working days, averaging approximately nine minutes per test case. After the proposed automated procedure was applied, total execution time dropped to one day — roughly 1.14 minutes per test case — an efficiency improvement of 87.5%. Beyond the time reduction, aligning automated testing practices with ISO/IEC 29119 produced a more disciplined testing workflow: formal test planning, structured monitoring, version-controlled test scripts, and standardized reporting replaced the informal, experience-dependent practices that had characterized the manual process.

The broader implication is that automation and standardization are more effective in combination than either is alone. Automation without structure produces speed without governance. Structure without automation produces documentation without efficiency. The procedure proposed here attempts to achieve both — and the results at PT. ABCD suggest the combination is viable in practice, even within an organization that had already reached CMMI Level 3 without formalizing its regression testing process. The study's limitations are real and should not be minimized. A single-site case study cannot establish generalizability, and

the evaluation's focus on execution time leaves questions about defect detection effectiveness and script maintenance costs unanswered. Maintaining a suite of 420 automated scripts across a frequently changing codebase carries ongoing costs that this study did not measure. Future work should examine the proposed procedure across multiple organizational contexts and attend more carefully to those dimensions — as well as to the integration of automated regression testing within CI/CD pipelines, where tests run automatically on every build and the value of a structured, standards-aligned procedure becomes even more pronounced.

References

Alshammari, M., & Harman, M. (2023). Data-driven approaches for regression testing optimization: A systematic mapping study. *Journal of Systems and Software*, *198*, 111569. https://doi.org/10.1016/j.jss.2022.111569

Ammann, P., & Offutt, J. (2017). *Introduction to software testing*. Cambridge University Press. https://doi.org/10.1017/9781107172012

Côrtes, C. T., Oliveira, S. M. J. V. D., Santos, R. C. S. D., Francisco, A. A., Riesco, M. L. G., & Shimoda, G. T. (2018). Implementation of evidence-based practices in normal delivery care. *Revista Latino-Americana de Enfermagem*, *26*, e2988. https://doi.org/10.1590/1518-8345.2177.2988

Fewster, M., & Graham, D. (1999). *Software test automation*. Addison-Wesley.

Gami, S., Katru, C., & Shah, K. (2025). Enhancing software reliability through automated CI/CD pipelines. *International Journal of Computer Applications*, *187*(1). https://doi.org/10.5120/ijca2025924785

Graham, D. (1999). *Software test automation: Effective use of test execution tools*. Addison-Wesley.

Gundecha, U. (2015). *Selenium testing tools cookbook*. Packt Publishing.

Hasnain, M., Pasha, M. F., Ghani, I., & Jeong, S. R. (2021). Functional requirement-based test case prioritization in regression testing: A systematic literature review. *SN Computer Science*, *2*(6), 421. https://doi.org/10.1007/s42979-021-00821-3

Humble, J., & Farley, D. (2010). *Continuous delivery: Reliable software releases through build, test, and deployment automation*. Pearson Education.

ISO/IEC/IEEE. (2013). *ISO/IEC/IEEE 29119-1: Software and systems engineering — Software testing — Part 1: Concepts and definitions*. International Organization for Standardization.

ISO/IEC/IEEE. (2013). *ISO/IEC/IEEE 29119-2: Software and systems engineering — Software testing — Part 2: Test processes*. International Organization for Standardization.

Jain, V. (2023). Continuous testing in CI/CD pipelines. *International Journal of Innovative Research and Creative Technology*, *9*(1), 1–7. https://doi.org/10.5281/zenodo.14883221

Kamiya, T., Kusumoto, S., & Inoue, K. (2002). CCFinder: A multilinguistic token-based code clone detection system for large scale source code. *IEEE Transactions on Software Engineering*, *28*(7), 654–670. https://doi.org/10.1109/TSE.2002.1019480

Lima, J. A. P., & Vergilio, S. R. (2020). Test case prioritization in continuous integration environments: A systematic mapping study. *Information and Software Technology*, *121*, 106268. https://doi.org/10.1016/j.infsof.2020.106268

Myers, G. J. (2006). *The art of software testing*. John Wiley & Sons.

Pan, R., Bagherzadeh, M., Ghaleb, T. A., & Briand, L. (2022). Test case selection and prioritization using machine learning: A systematic literature review. *Empirical Software Engineering*, *27*(2), 29. https://doi.org/10.1007/s10664-021-10054-7

Panichella, A., Kifetew, F. M., & Tonella, P. (2017). Automated test case generation as a many-objective optimisation problem with dynamic selection of the targets. *IEEE Transactions on Software Engineering*, *44*(2), 122–158. https://doi.org/10.1109/TSE.2017.2663435

Parekh, K., & Saxena, S. (2025). Automating regression testing: Optimizing efficiency and coverage in complex software systems. *International Journal for Research Publication and Seminar*, *16*(2), 135–142. https://doi.org/10.36676/jrps.v16.i2.58

Shahin, M., Babar, M. A., & Zhu, L. (2017). Continuous integration, delivery and deployment: A systematic review on approaches, tools, challenges and practices. *IEEE Access*, *5*, 3909–3943. https://doi.org/10.1109/ACCESS.2017.2685629

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

Zhang, J., Liu, Y., Gligoric, M., Legunsen, O., & Shi, A. (2022, May). Comparing and combining analysis-based and learning-based regression test selection. In *Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test* (pp. 17–28). https://doi.org/10.1145/3524481.3527230.