Regression Automation: Methods to Reduce Testing Time and Improve Product Quality

Mohnish Neelapu

Domain: QA Automation

Abstract:
In software development, regression testing is a crucial part to ensure that newly introduced changes do not negatively affect workings of already existing features. However, manual regression test is cost intensive, time consuming, tedious and prone to human errors and actually become bottlenecks in the software development cycle. Automated regression testing is used as a way to improving the efficiency, accuracy, and detection of defects with overall testing time reduced. Then it analyzes various automation techniques such as test prioritization, self healing mechanisms and continuous integration and deployment (CI/CD) integration that help reduce the need for software tests and increase product quality. Automation study is performed and the impact of automation on critical performance metric such as defect detection rate, test execution time, regression failure rate and code coverage is studied. This research presents a comparative analysis of how the test cycle can be accelerated by automation, defects are detected with a higher accuracy, and the code coverage can be enlarged through automation. Execution time is significantly reduced and failure rates are significantly reduced with experimental findings, as are defect detection efficiency. Automated regression testing strategies adopted by organizations can help not only optimize the test workflows that become more expeditious and prevent product meltdown by further accelerating the software release cycle, but also reduce the maintenance overhead of software testing which further increase the quality and reliability of software. The data which emerge from this research only confirm the essence of automation in present day software development, giving teams a hand on how to lessen the testing time, keep it maintainable, and improve software reliability in agile and rapidly paced environments.

Keywords: Regression Testing Automation, Software Quality Assurance, Test Execution Optimization, Defect Detection Efficiency, Continuous Integration and Deployment (CI/CD).

I. INTRODUCTION

Regression testing is an important part of the software development process as it guarantees that added changes in the post do not affect the existing functions [18]. However, with updates, bug fixes and feature enhancements all becoming part of software evolution, the problem of acquiring and keeping software stable and reliable becomes more and more difficult [17]. While effective, the traditional manual regression testing is extremely time consuming, laborious and should be performed by a human [7]. For complex applications, the number of test cases needed can be simply overwhelming for testers and often it is simply infeasible to expect that these test cases are covered within project deadline. Moreover, manual execution lacks consistency, and thus there are variations in the test results across different test cycles. Such inefficiencies in the software development cycle enhance the cost, extend the time and increase the risk of undetected defects from entering production [8]. Thereby, businesses have issues in providing high quality software at a fast speed. With growing complexity of modern software systems, organizations now are being pushed to look for alternative test approaches that optimize efficiency, reduce human involvement as well as improve overall product quality [9] [19].

In order to tackle these problems, automated regression testing is a triumphant solution that turns testing into an automated process [20]. Testing speed, accuracy and repeatability is increased by automating them with no man dependency. Performed by automated test scripts, thousands of test cases can be executed a lot

faster than a manual tester would be able to and thus faster detection and resolution of defects [10]. Automation frameworks also include support for risk based test selection through which the organizations can select the test cases based on the critical application components and new code changes. Parallel execution further accelerates test execution time, improving the test execution time and facilitating rapid feedback loops for the developers. Also, integration with CI/CD pipelines eases the software releases by automating the regression tests after each code commit [11]. In such modern Agile and DevOps environments where software updates are constant; automation becomes a necessity because of these advantages. With automation in place, organizations can increase the efficiency and optimize the utilization of resources as well as achieve faster time to the market and stable software [12][13].

While efficient regression automation strategies are evident, they come with many challenges which the organization needs to tackle. There is an initial cost into the infrastructure, tools, and enough skilled personnel that it can be significant [16]. It also requires maintenance of automated tests both in terms of effort and of maintaining the tests so that failures due to outdated selectors or deprecated functions do not occur with every application change [14]. Test automation frameworks must also be evaluated by ensuring their suitability regarding the application architecture and the development workflow. Automated regression testing solutions provide self-healing mechanisms as well as intelligent test selection, but their adoption depends on organizational readiness and technical feasibility [15]. This research studies methodologies that help automate regression testing to improve software quality and testing efficiency. The paper describes the advantages, challenges, and practical implications of automated regression testing in modern software development environments, outlining best practices for enhancing test execution and defect detection.

II. LITERATURE REVIEW

In recent years, a number of researchers explored different regression automation techniques in order to improve the efficiency of software testing. In CI environments, Elsner et al. [1] developed Build System Aware Multi language Regression Test Selection method. Using this approach, dependencies across several programming languages are considered to optimize test selection with higher execution time, and reduced computational overhead. However, integrating into different CI workflows is difficult because of the complexity of analyzing diverse build systems. In [2], Nițescu et al. proposed an automation framework for integrating CI into software development that concentrates on automated test execution in fast paced DevOps environment. By adopting their method, automated tests become seamlessly integrated into CI pipelines and thus help improving software reliability. In dynamic application environments with frequent updates, the main challenge is in maintaining automation scripts, particularly as in a dynamic application environment, changes occur very often. A Component-Based Requirements Prediction and Regression Testing Aspects Framework as proposed by Ali et al. [3] are presented, to enhance the efficiency of test case selection by predicting the change impact on software. On this approach, it minimizes the need of redundant test executions and enhances the fault detection rate. But depending on the component dependency mapping, the model may be mistaken which introduce inconsistencies in case of lack of correct documentation. In Krushna and Gopinath [4] developed an Agile Test Automation framework for web applications based on TestNG, Random Integration Algorithm of Machine Learning, has been developed. Using this approach leads to improved accuracy of the tests and prediction of response time, resulting in accurate identifying of defects in the agile environment. However, the model considerably improves test reliability while simultaneously posing a significant roadblock in dealing with the flexibility of assembling web applications, due to which the model needs to be retrained very often. Othman and Zein [5] developed Model Based Testing (MBT) for web applications where test cases are auto generated from system models. The advantage of this technique is significant reduction in human effort needed for test creation as well as complete test coverage. However, its effectiveness is limited in accuracy by the accuracy of the system models, which can be time consuming to develop for complex applications.

A. Challenges in Regression Automation

Despite improvements in the automation of regression, there are several challenges that are still not addressed:

- Currently, the test selection methods need to be scaled to large applications with constantly changing codebases, which causes inefficiency in the regression testing.
- Frequent updates are necessary for automation frameworks to adjust to changing application structures, which raises maintenance costs and resource usage.
- Predictive test selection and self-healing mechanisms are still difficult to implement because research on their viability and efficacy is still ongoing.
- Even though CI/CD pipelines improve automation, a lot of research concentrates on execution efficiency without taking into account how well they integrate with DevOps workflows.
- It is still difficult to strike the ideal balance between test coverage and execution time because reducing execution time frequently results in lower fault detection rates.

III. PROPOSED METHODOLOGY FOR REGRESSION AUTOMATION

In the proposed methodology for regression automation, an experimentation approach is used to explore a number of automation techniques in real world testing situations. Automation tools and frameworks are compared to try and ascertain their leadership and capacity in decreasing test execution time with the increase of product quality. Secondly, the methodology is validated with industry case studies, to prove the practical applicability of the methodology in the entire software development environment. It based data collection on performance benchmarks, the best practices into the industry, and real world test suites to get a full evaluation. Qualification of the automation tools under review include Selenium, Cypress, TestNG, and JUnit all tools with the scope for script execution, test case management and defect detection having wide software application in the industry. In the experimental setup set up a dedicated testing environment where a lot of different versions of the software, automation frameworks, and CI/CD integrations are running in parallel and picking a subset of tests to run and execute dynamically for better efficiency. Finally, our evaluation criteria are test execution time, defect detection rate and, resource efficiency and serves as measurable insights into effectiveness of the strategy. The experiments on real world applications are conducted to ensure practical relevance and to show the impact of regression automation in a number of testing scenarios.

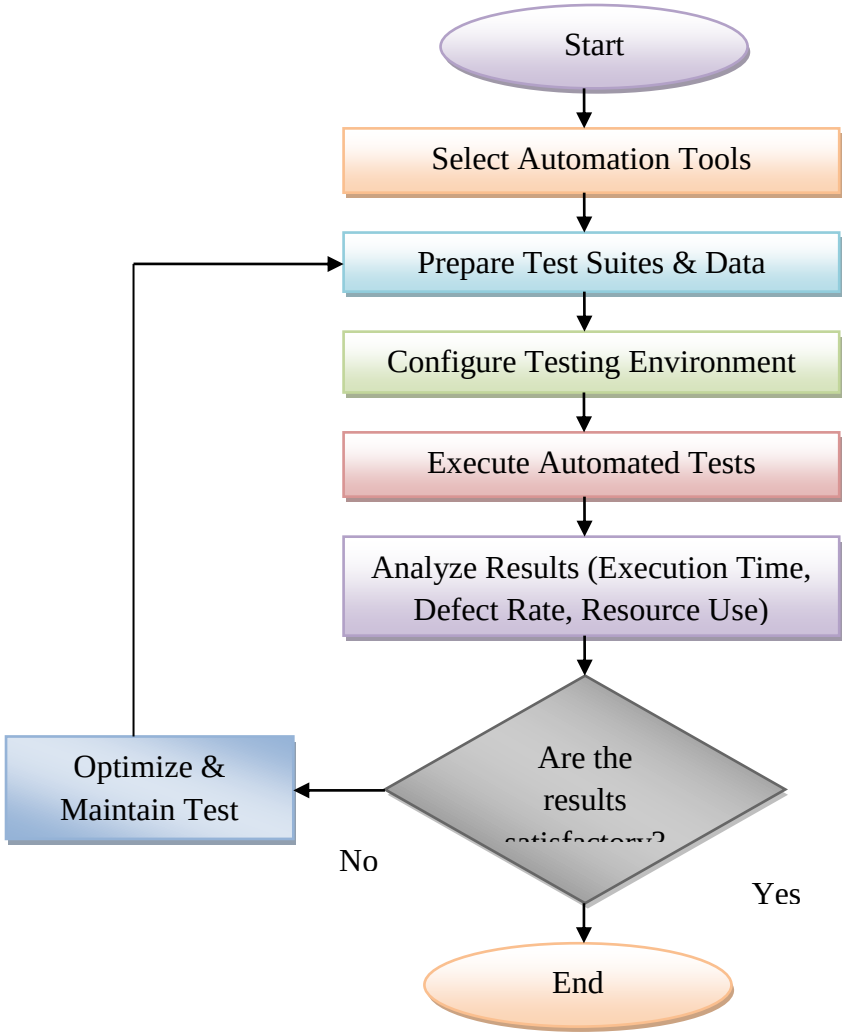

Fig. 1. Proposed Workflow for Regression Automation.

A. Methods for Reducing Testing Time

Automatic regression runs need efficient strategies to ensure software reliability and minimize the accumulation of regression run time. In this section, we share key methodologies for speeding up test execution speed, augmenting defect detection efficiency and getting the integration of automation worked into a smooth development cycle. The proposed approaches are about prioritization, parallel execution, self healing test mechanism, and integration with CI/CD pipeline.

1) Test Prioritization Strategies

Risk based selection and historical defect analysis are modern techniques to handle the automation driven approach of test prioritization. Based on this, risk based prioritization will select test cases according to the factors of code changes, defect history and business impact first. In addition, automation frameworks allow test execution to be organized dynamically by previously failed test patterns and execution times. Thus risk based prioritization along with the use of automated structured approach to prioritization can help team's utilized time to execute without compromising defect detection accuracy.

2) Parallel Execution and Distributed Testing

Parallel execution and distributed testing are one of the most effective ways in which the regression testing time can be reduced. With parallel testing, we distribute the workload to multiple environments, devices or cloud instances and the workload is executed in parallel. Concurrent test execution is made possible with Selenium Grid, containerized test environments (Docker) and cloud platform based testing platform (AWS Device Farm, BrowserStack). More efficiency is added by the distributed testing which utilizes multiple

machines or cluster to execute multiple test cases simultaneously. It significantly shortens the test execution cycle, especially in big scale applications that need fast updates.

3) Automated Test Maintenance and Self-Healing Mechanisms

Programs with changing UI components and backend logic need automatic test maintenance in dynamic software environment. The increase in maintenance effort in maintaining script reliability means traditional regression suites require frequent updates. In order to tackle this concern, automation frameworks use self healing features by automating test scripts when application components are changed. These mechanisms use defined fallback locators, structured element identify techniques, and reliable error handling to detect any modifications like XPath or CSS selector updates. With the help of structured self healing mechanisms, teams ensure that long term efficiency and sustainability of the automation process is maintained through manual intervention as much as possible.

Pseudocode 1: Self-Healing Mechanism for Automated Regression Testing

```
function self_healing_test(test_case):
try
        execute(test_case)
except ElementNotFound:
        new_locator = find_updated_locator(test_case)
if new_locator:
        update_locator(test_case, new_locator)
execute(test_case)
else:
        log_failure(test_case)
```

This is pseudocode that exhibits a dynamic method to where alternative locators are generated and validated by it. The test case is re-executed with updates if a new, valid locator is found so as to minimize impact to test automation workflows.

4) Optimizing CI/CD Pipeline Integration

To enable speeding up the software delivery cycles while maintaining high quality, it is to be integrated seamlessly with the CI/CD pipeline. An important part of continuous integration is to have automated testing trigger as early as possible to detect defects in the early stages of the development process, so you can count on stability when it is being deployed. Automated tests can be executed on every code commit, as a way to identify regressions as early as possible in introducing the new code, and to optimise test scheduling to the optimal mix of execution time and coverage, all of this can be achieved. With embedded automated regression testing into DevOps workflows, development becomes faster, reliability of the software improved, and rapid product releases not affecting quality are achieved.

B. Improving Product Quality through Automation

Automated regression testing is critical in improving the software quality by checking if any defects are included in the new updates. Regression automation helps provide the product stability, consistency, and reliability while expunging manual intervention, and speeding up the test execution. This section explains key steps of how to bring the power of automation over product quality, including test coverage and how to keep the product quality running continuously with awareness of certain quality metrics.

1) Enhancing Test Coverage in Regression Automation

The need for test coverage is critical in order to ensure software reliability because an insufficient amount will not catch any potential defects. Regression automation allows for the test coverage to be increased by systematically running many test cases on a wide range of environments and configurations. Due to this, automated regression frameworks use dynamic techniques of test selection and prioritization to augment test coverage for the most critical functionalities. It ensures that kernel risk areas of software undergo rigorous

testings, therefore minimising the possibility of regression failures in production. Furthermore, the automatic testing also gives cross platform validation, meaning that it has the same performance whether your device is Android, iOS, Chrome, Browser, or whatever your platform or operating system may be currently. Structured test coverage strategies can be implemented by the organizations to increase the likelihood of detecting defects which lowers the risks of software updates.

2) Feedback Mechanisms and Continuous Monitoring

Software performance is checked for in automated regression testing, and it offers real time feedback to the development teams to detect defects early on in software lifecycle. This should be a continuous observation of test execution to track failure pattern, trend of stability and deviation of performance throughout the time. Good feedback mechanisms involve generating a detailed test report where pass or fails rates, error log and execution trends helps a team intelligently review test outcomes. Also, defect tracking tools integration makes these failures logged automatically and categorized, simplifying debugger and troubleshooting. Additionally, stakeholder feedback loops promote working together between developers, testers and project managers to create test strategies in a real-time environment. Regression automation helps to increase the quality of software, minimize defects leak into production and maximally reliability of applications by integrating real time monitoring and structured feedback mechanisms. Fig. 2 illustrates the feedback Mechanisms and Continuous Monitoring in Regression Automation

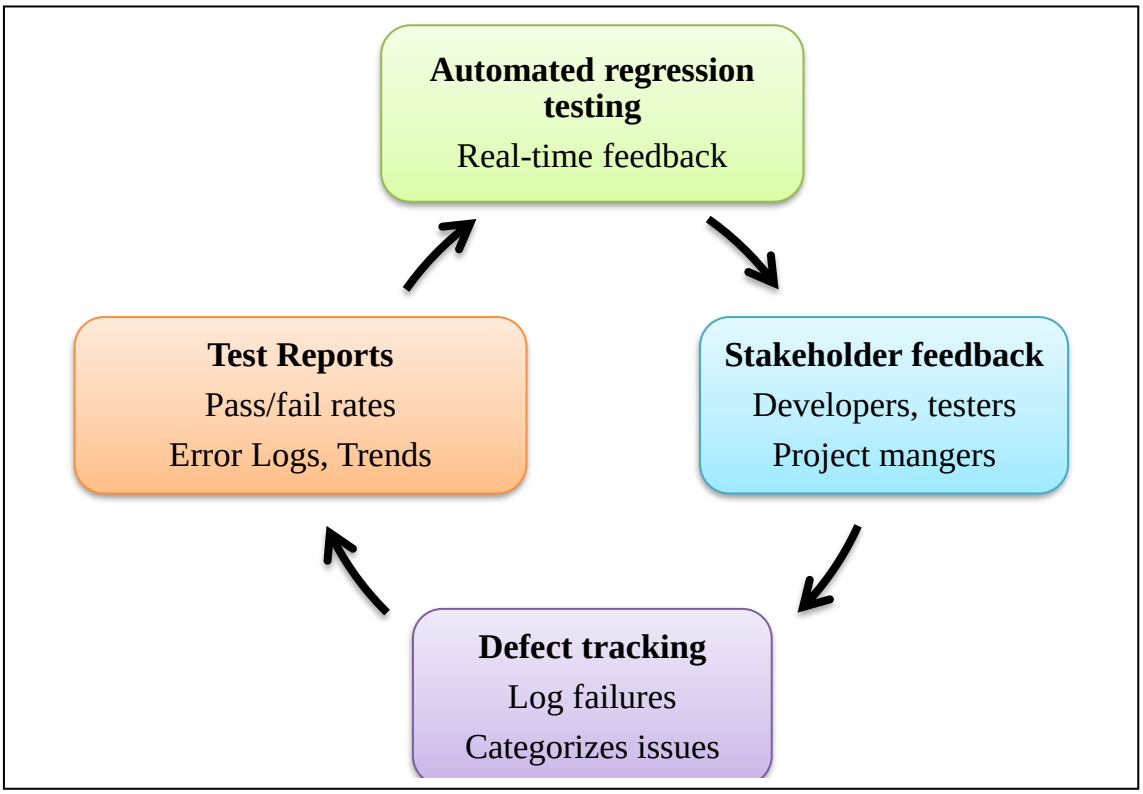

Fig. 2. Feedback Mechanisms and Continuous Monitoring in Regression Automation.

3) Quality Metrics for Automated Regression Testing

To perform automated regression, organizations define well defined quality metrics which quantitative provide insights about the efficacy of automated regression in regard to the test and defect detection efficiency. The Defect Detection Rate (DDR), being one the key metrics, is the percentage of defects identified through automated regression tests and prevents critical defects from happening at the end of the development cycle. Another important indicator is Test Execution Time, because it measures the testing process efficiency by how long it takes to execute automated test cases in order to optimize it. Furthermore, the Regression Failure Rate shows how often new updates lead to test failures, providing assurance on the software stability and the effect of code change. Another crucial one is Code Coverage, which is an important metric that indicates whether application code is successfully tested to ensure complete test for

key functionalities with the help of regression tests. By studying these metrics, regression automation strategies of the organizations can be fine tuned to increase defect detection efficiency, minimize test execution speed and maximize software quality.

IV. RESULT AND ANALYSIS

The Results Section reports on regression automation experiment results aimed at testing efficiency and product quality improvements. Quantitative performance comparisons and graphic and tabular data to show important insights should be included in this section.

A. Performance Comparison of Regression Automation

Experiments were conducted to measure the impact of automation on testing time and defect detection for different automation framework (Selenium, Cypress, TestNG, etc.). Results show in table I that the utilization of automated regression testing leads to a great decrease in execution time without incurring a decrease in defect detection accuracy.

TABLE I- REGRESSION AUTOMATION PERFORMANCE METRICS

Metric	Before Automation	After Automation	Improvement (%)
Test Execution Time (mins)	120	45	62.5%
Defect Detection Rate (%)	78	92	17.9%
Regression Failure Rate (%)	15	7	-53.3%
Code Coverage (%)	65	85	30.8%

B. Reduction in Test Execution Time

Execution time of a test cycle in a regression testing is the total time it takes to execute a test cycle for a set of test cases. One of the metrics that needs to be considered in evaluating an automated testing framework's efficiency is this. Test execution time is an important metric which should be reduced to accelerate the software delivery cycles without compromising on the product quality. It has been seen from the figure 3 tested execution time reduces significantly under the support of automation. Automated testing was quicker to run through parallel processing and a scripted basis, as compared to manual testing which took longer. Particularly, this improvement increases efficiency, speeds up the delivery of software, and makes sure that the software is fully regression tested while maintaining the quality.

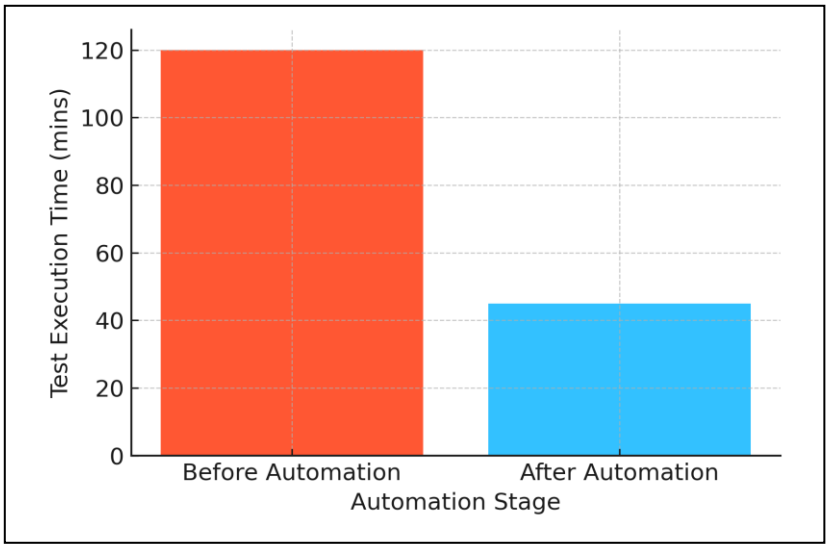

Fig. 3. Test Execution Time Comparison Before and After Automation.

C. Defect Detection Efficiency

Regression testing automates software issues and the early identification of defects improves the defect detection. Better software stability and reliability can be achieved with this improvement. It is the defect detection rate, which can be defined as the percentage of defects found by automated testing as compared to manual efforts. We can see a consistently increasing in defect detection efficiency by analysis of several test cycles, this shows how automating can

reduce unexamined errors. This fig. 4 shows the increase in defect detection rate over multiple test cycles.

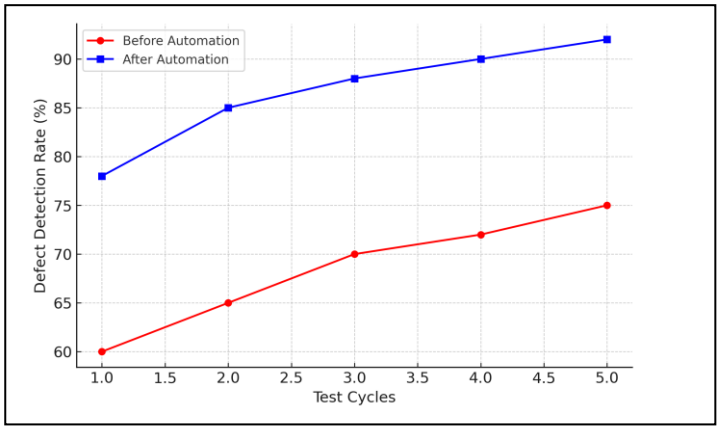

Fig. 4. Defect Detection Rate over Test Cycles.

D. Stability and Regression Failure Trends

One of the most important goals of regression automation is to reduce regression failure due to new code change. Failures are high in manual regression testing and often do not find all defects efficiently. Automation of tests runs the scripts the same way repeatedly, finding regression early and having fewer failures. The table II below compares the regression failure rate before and after automation, demonstrating a significant reduction in failure rates, thereby improving software stability and reliability.

TABLE II- COMPARISON OF FAILURE RATES BEFORE AND AFTER AUTOMATION ACROSS TEST CYCLES

Test Cycle	**Failure Rate Before Automation (%)**	**Failure Rate After Automation (%)**
Cycle 1	18	9
Cycle 2	15	7
Cycle 3	12	6

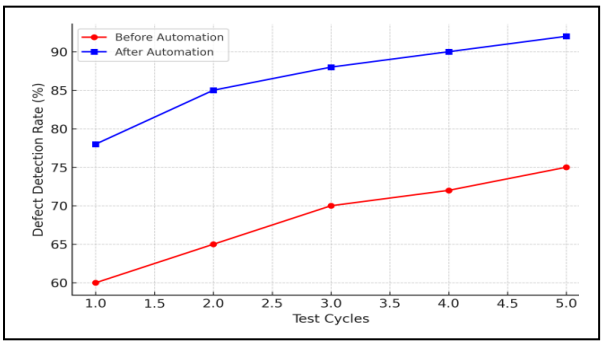

Fig. 5. Regression Failure Rate Reduction Trend.

F. Code Coverage Enhancement

Eliminating manual regression testing greatly increases the area of code coverage in relation to the size of the computer code being tested. More code coverage also results in more functionality verified that would be missed by defect detection in critical modules. The increase of code coverage to automate and before automation is shown in this figure 6, showing that automation increases in software reliability.

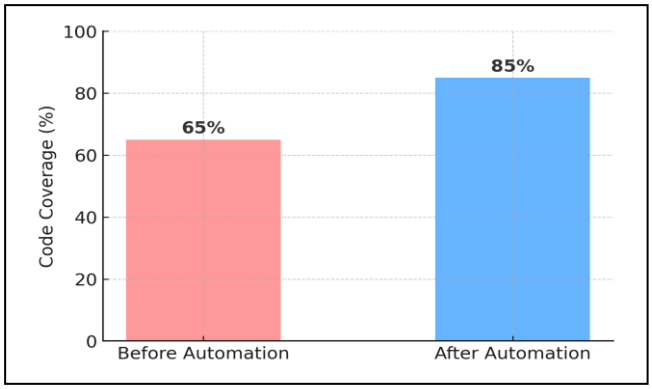

Fig. 6. Code Coverage Before and After Automation.

V. CONCLUSION

In this study, automated regression testing is important, as it speeds up the testing time and keeps the bug free of production products. For organizations, the replacement of labor intensive labor intensive manual testing by structured automation leads to faster test execution leading to higher defects detection rate and better quality of software validation. It is found that failure rates can be minimized and all test cases to be covered with test prioritization, automated test maintenance, and self healing mechanisms. Despite the initial investment in tools, infrastructure, and skilled resources, the long-term advantages of automation far outweigh its challenges. This improves the organizations' efficiency, and scalability, and easier integration with CI/CD pipelines in order to do continuous testing and frequent software releases with no compromise of stability. In addition it puts forward the need of structured test suite design and robust maintenance strategies to keep effectiveness of automation over time. Future work can focus on improving the automation frameworks, test suite selection strategy as well as maintainability to further improve the efficiency of the regression testing. In reality, the organizations that can strategically promote automation can speed up their development cycle, improve product reliability, and remain competitively aggressive in the rapidly changing software field.

REFERENCES:

[1] D. Elsner, R. Wuersching, M. Schnappinger, A. Pretschner, M. Graber, R. Dammer, and S. Reimer, "Build system aware multi-language regression test selection in continuous integration," *In Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, 2022, May , pp. 87-96.

[2] T. A. Nițescu, A. I. Concea-Prisăcaru, and V. Sgârciu, "Test Automation for Continuous Integration In Software Development".

[3] S. Ali, Y. Hafeez, M. Humayun, N. Z. Jhanjhi, and R. M. Ghoniem, "An Aspects Framework for Component-Based Requirements Prediction and Regression Testing," *Sustainability*, vol. 14, no. 21, pp. 14563, 2022.

[4] V. V. Krishna, and G. Gopinath, "Agile Test Automation For Web Application Using Testng Framework With Random Integration Algorithm In Machine Learning To Predict Accuracy And Response Time On Automated Test Results," *Journal of Theoretical and Applied Information Technology*, vol. 100, no. 16, 2022.

[5] R. Othman, and S. Zein, "Test Case Auto-Generation For Web Applications: A Model-Based Approach," *In 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT). IEEE*, 2022, October, pp. 18-25.

[6] A. Geistanger, K. Braese, and R. Laubender, "Automated data analytics workflow for stability experiments based on regression analysis," *Journal of Mass Spectrometry and Advances in the Clinical lab*, vol. 24, pp. 5-14, 2022.

[7] R. R. Karn, R. Das, D. R. Pant, J. Heikkonen, and R. Kanth, "Automated testing and resilience of microservice's network-link using istio service mesh," *In 2022 31st Conference of Open Innovations Association (FRUCT)*. *IEEE*, pp. 79-88, 2022, April.

[8] L. Yin, H. Zhang, Z. Tang, J. Xu, D. Yin, Z. Zhang, and X. Liu, "rMVP: a memory-efficient, visualization-enhanced, and parallel-accelerated tool for genome-wide association study," *Genomics, proteomics & bioinformatics*, vol. 19, no. 4, pp. 619-628, 2021.

[9] M. Felderer, and R. Ramler, "Quality assurance for AI-based systems: Overview and challenges (introduction to interactive session)," *In Software Quality: Future Perspectives on Software Engineering Quality: 13th International Conference, SWQD 2021, Vienna, Austria, January 19–21, 2021, Proceedings Springer*, vol. 13, pp. 33-42, 2021

[10] F. Almeida, J. Simões, and S. Lopes, "Exploring the benefits of combining devops and agile," *Future Internet, International Publishing*, vol. 14, no. 2, pp. 63, 2022.

[11] M. Bagherzadeh, N. Kahani, and L. Briand, "Reinforcement learning for test case prioritization," *IEEE Transactions on Software Engineering*, vol. 48, no. 8, pp. 2836-2856, 2021.

[12] F. S. Ahmed, A. Majeed, T. A. Khan, and S. N. Bhatti, "Value-based cost-cognizant test case prioritization for regression testing," *Plos one*, vol. 17, no. 5, pp. e0264972, 2022.

[13] D. Chicco, M. J. Warrens, and G. Jurman, "The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation," *Peerj computer science*, vol. 7, pp. e623, 2021.

[14] M. Waseem, P. Liang, M. Shahin, A. Di Salle, and G. Márquez, "Design, monitoring, and testing of microservices systems: The practitioners' perspective," *Journal of Systems and Software*, vol. 182, pp. 111061, 2021.

[15] A. Tyagi, "Intelligent DevOps: Harnessing Artificial Intelligence to Revolutionize CI/CD Pipelines and Optimize Software Delivery Lifecycles," *Journal of Emerging Technologies and Innovative Research*, vol. 8, pp. 367-385, 2021.

[16] M. Neelapu, "Impact of Cross-Functional Collaboration on Software Testing Efficiency," *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*, vol. 10, no. 6, Dec. 2022, Article 232322.

[17] M. Neelapu, "Hybrid Testing Frameworks: Benefits and Challenges in Automation," *International Journal for Multidisciplinary Research*, vol. 4, no. 6, Nov.–Dec. 2022.

[18] D. Elsner, F. Hauer, A. Pretschner, and S. Reimer, "Empirically evaluating readily available information for regression test optimization in continuous integration," *In Proceedings of the 30th ACM SIGSOFT international symposium on software testing and analysis*, pp. 491-504, 2021, July.

[19] Mohnish Neelapu, "Impact of cross-functional collaboration on software testing efficiency," *International Journal of Innovative Research in Engineering & Multidisciplinary Physical Sciences*, vol. 10, no. 6, 2022, Article 232322.

[20] Mohnish Neelapu, "Enhancing Agile Software Development through Behavior-Driven Development: Improving Requirement Clarity, Collaboration, and Automated Testing ESP," *Journal of Engineering & Technology Advancements*, vol. 3, no. 2, pp. 153-161, 2023.