Online-Journals.org

JFSE Journal for Future Society and Education

JFSE | eISSN: 3078-6010 | Vol. 2 No. 4 (2025) | OPEN ACCESS

https://doi.org/10.3991/jfse.v2i4.59235

PAPER

Hybrid Automated-Manual Testing for Enhanced DevOps Pipelines

Samar Abbas[1](✉), Saman Masood[1], Sidra Tahir[1], Muhammad Azhar[1], Ahthasham Sajid[2], Sabitha Banu[3]

[1]PMAS – Arid Agriculture University, Rawalpindi, Pakistan

[2]Air University, Islamabad, Pakistan

[3]PSGR Krishnammal College for Women, Coimbatore, India

abbassammar549@gmail.com

ABSTRACT

DevOps is a paradigm shift in software development today, aimed at the rapid delivery of software through a task-oriented approach to work and the integration of development and operations teams. Software testing is one of the most important and challenging steps in the DevOps pipeline because it should be effective and stable without being slow. To improve the process of software testing among DevOps professionals, this paper suggests a semi-automatic testing approach as a more viable solution between the extremes. The semi-automated methodology involves automation of processes that are tedious and time-consuming (such as unit testing, regression testing, and continuous integration), but human inspection in processes that require critical thinking, business domain knowledge, and exploratory insight (such as UI validation, business logic, and end-user behavior simulation). The combination creates the ability to do faster feedback loops, reduced test maintaining burden, improved test coverage, and quality assurance (QA) in general. The study utilizes a combination of case study evaluation, the implementation of the tools, and the reaction of the practitioner to the survey to gauge the performance of semi-automation in the DevOps context. The core performance indicators assessment of performance includes reduced test cycle time, defect leakage rate, productivity of the team, and the rate of deployment. The findings indicate that semi-automatic solutions result in radical improvements of test efficiency, scalability, and flexibility, especially in groups with dynamic codebases and limited automation budgets. The proposed study will integrate an affordable yet quality testing model that is dynamic enough to keep up with the DevOps and the advancement of the ideal balance between human judgment and automation. The method described here does not only enhance the effect of testing as an undertaking but also does the more universal DevOps objectives of quicker delivery, quality, and better teamwork.

KEYWORDS

software testing, semi-automated testing, DevOps practices, testing automation, quality assurance (QA), manual and automated testing, software delivery, DevOps testing challenges

Abbas, S., Masood, S., Tahir, S., Azhar, M., Sajid, A., Banu, S. (2025). Hybrid Automated-Manual Testing for Enhanced DevOps Pipelines. *Journal for Future Society and Education (JFSE)*, 2(4), pp. 66–77. https://doi.org/10.3991/jfse.v2i4.59235

Article submitted 2025-09-03. Revision uploaded 2025-10-13. Final acceptance 2025-10-27.

1 INTRODUCTION

A typesetting team will do the final touch on your manuscript; you don't need to. Over the last ten years, your final touch on the manuscript will be done by a typesetting team; you need not worry about the rest. Over the last ten years, DevOps has transformed the software development process to bring together development (Dev) and operations (Ops) teams to shorten cycle times and enhance delivery reliability [1]. The main DevOps practices are continuous integration (CI) and continuous deployment (CD), in which the version control automatically creates code changes, which are then automatically tested and automatically deployed to a deployment system, using a series of automation pipelines and test suites [2], [3]. CI enables defects to be detected early and can be dealt with faster, CI encourages maintainable, modular codebases; and CD enables the software to become usable when it is needed and at a fast pace [1], [4].

DevOps puts some of its complexities on testing practices, although it has positive impacts. Sometimes the idea of end-to-end automated pipelines, which is desirable due to its speed and reproducibility, will conflict with its boundaries. Automated testing is also effective at running repeatable unit and integration and regression scale tests but has high initial costs and maintenance costs, and can have flaky or brittle test scripts, particularly when it comes to GUI-based testing [5], [6], [7].

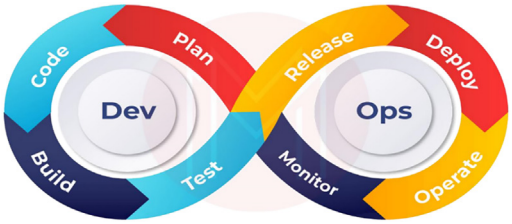

Fig. 1. DevOps life cycle

Conversely, manual testing in particular in such areas as exploratory, usability, localization, or domain-specific logic offers fine-grained information that cannot be obtained under automated methods [10], [11]. Human testers are most effective in finding context-based bugs, validation, and dealing with edge cases, which robotic scripts overlook. Nonetheless, manual operations are causing bottlenecks: they are laborious, uneven, and slow, making it impossible to release fast-paced DevOps releases [10], [12]. Manual regressions are more time-consuming and costly to execute and demand more expertise, which is less reproducible and less expensive in the long term [10], [12].

Comparisons of both extremes show that automation brings in the benefit of scale at the cost of fragility and high maintenance, whereas manual testing brings in depth at the cost of speed and repeatability. As an illustration, a detailed empirical investigation of visual GUI test suites in Siemens and Saab showed that despite the fact that automation led to a decrease of defect leakage and development expenses, continuous maintenance was substantial, particularly when test cases were introduced either infrequently or in bulk [8]. In addition, organizations that have implemented CI/CD pipelines have also indicated pipeline blockages through flaky tests, which have compromised the velocity of development and the confidence of the developers in the test results [6], [9].

The way forward in this tension is to develop a semi-automated test methodology that will involve automated efficiency with a human touch. Deterministic and repeated testing tests like unit, smoke, and regression tests in this hybrid model are automated and are integrated into CI/CD pipelines. In the meantime, exploratory, usability, and UI validation, security edge-case testing and complex scenario analysis are all done manually.

This approach allows giving quick and prompt feedback without losing the manual richness in case of context-sensitive situations [13], [6], [10]. The time-oriented defect detection, resource utilization, and maintainability are managed by semi-automation by aligning shift-left in the dev lifecycle provided earlier [6], [13], [14].

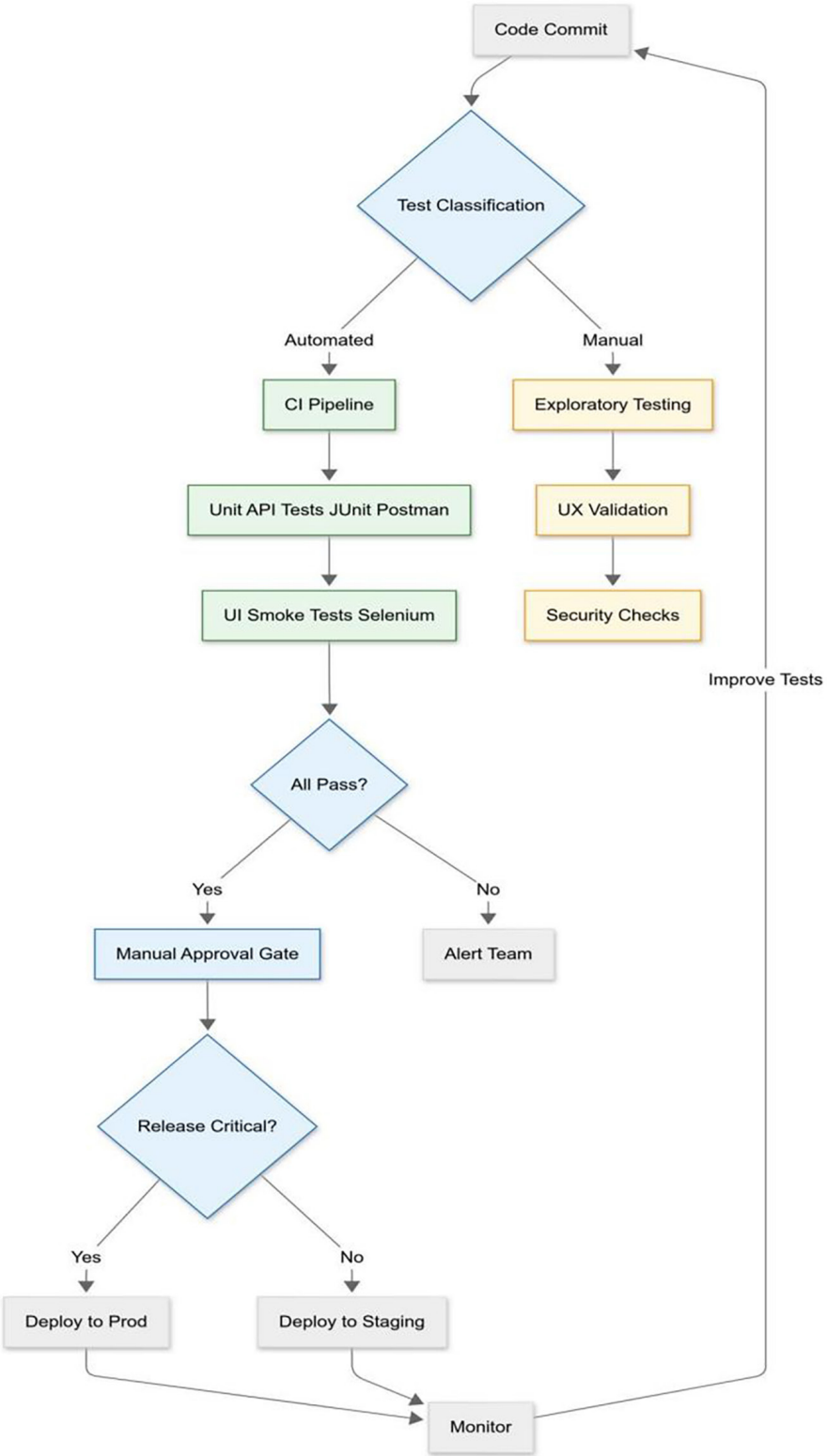

Fig. 2. Work flow diagram of hybrid testing

The semi-automated model does not only ease the human resource to do the repetitive work but also minimizes the maintenance of automation scripts. Having a smaller and more specific scope, automated tests are leaner and more robust and are not prone to brittleness or flaky behavior [8], [14]. Where their expertise is most effective UX, exploratory testing, and critical business logic testing, manual testers can work, and where they have a stable foundation, automated tests can provide.

A number of cases in the industry have favored this hybrid approach. An example is semi-automated testing implemented in Siemens and Saab that allowed the company to have shorter maintenance cycles and higher ROI than fully automated or manual approaches [8]. In another instance, DevOps teams operating under a hybrid regimen that combined automation based on regression and manual investigators enhanced speed and completeness without pipeline failures of flaky scripts [6], [10], [14]. Tasks such as containerization, test running in parallel, and orchestration optimization (e.g., Docker, Kubernetes, Jenkins) optimize semi-automated workflows' throughput [3], [10].

But the semi-automated model has its own issues. Teams must have explicit rules for when tests are automated and when to leave them in the human area [6], [10], [12]. Creating a proper test taxonomy requires inter-role coordination by the devs, testers, and ops to assess risk, number of runs, and automation potential. Infra decisions require careful planning: test environments must be containerized and scalable, reproducible, and resilient while introducing minimal environmental variance for test outcomes [3], [10]. Role migrations also require training, cultural acceptability, and potential upskilling of testers into role blends (e.g., SDET) [12], [6]. Finally, smooth integrations between CI/CD pipelines, test management tools, reporting tools, and operations workflows are required but are muddled [3], [6].

Given these complexities, our research seeks to answer several critical questions:

- Which testing activities best suit automation, and which still require manual execution?
- How should a semi-automated test taxonomy or model be formulated?
- What infrastructure and tooling architectures best support semi-automated pipelines?
- What metrics for test cycle time, defect leakage, maintenance overhead, test coverage, cost, and team satisfaction should be tracked to evaluate the hybrid model?

To address these questions, a multi-method approach is undertaken: a literature survey to consolidate current knowledge from case studies, systematic reviews, and industry reports; a prototype semi-automated framework implemented with common DevOps tools (e.g., Jenkins, Docker, Selenium, JUnit, and Postman); and empirical validation through pilot deployments, quantitative measurement, and practitioner feedback. The data points such as time-to-feedback, failure rates of script, maintenance effort documentation, and qualitative measurements of satisfaction will be collected and analyzed. The research here wants to create a proven framework that will aid in guiding DevOps organizations towards adopting this pragmatic path while providing software quality without the lost velocity.

The later sections of the paper are organized in this manner: Section 2 comprises the detailed literature survey of the prevailing software testing and semi-automated DevOps practitioner methodologies and their fallibilities. Section 3 defines the

research gap identified and creates the problem statement driving the need for new approaches. Section 4 explains the proposed approach in the form of the software testing and automated DevOps practitioner methods. Section 5 discusses the experimentation setup, datasets employed, evaluation metrics, and details of implementation. Section 6 demonstrates and discusses the results, making a comparison between the proposed model's performance and the baselines. Finally, Section 7 summarizes the paper by giving the key results, practical contributions for DevOps practitioners, research limitations, and future research avenues for DevOps practitioners.

2 LITERATURE REVIEW

Mohan and Ben Othmane [1] offer a detailed work published in the *International Journal of Artificial Intelligence and Machine Learning in Engineering* regarding automated testing strategies for DevOps. Integration of CI/CD pipelines and test environments that are containerized are given priority by them, which results in substantial improvements in feedback loops and code quality. Higher setup cost and maintenance overhead are also the concerns noted by them.

Shahin et al. [2] conducted a systematic literature review (SLR) for continuous CI, CD, and deployment. They point to semi-automated testing as a prime area where they suggest using automation for low-context activities and by-hand verification for exploratory or UI-dominated cases. Their taxonomy provides basic direction for hybrid DevOps testing.

In their experience report, Wang et al. [3] describe the test automation process improvement (TAPI) implementation in an F-Secure DevOps team. They identify that gradual adoption, end-to-end team participation, and concise telemetry are the key success factors that result in rapid release, improved productivity, and increased test efficiency.

Auerbach et al. [4] explored the concept of continuous testing by establishing the guiding principles thereof and advocating its role in risk analysis for all development stages. According to these authors, early investment in automated testing reduces integration problems and aids the "shift-left" culture.

Alégroth et al. [5], in their empirical case study of visual GUI testing for Siemens and Saab companies, state that small-scale maintenance on a regular basis is less expensive than big batch updates. Although the automation lowers the expenses for manual testing, maintenance remains a significant share of total V&V efforts.

Mudadi and Lotriet [6] explore the organizational impacts of embracing DevOps in a multinational corporation in South Africa. The research highlights the importance of test automation in brokering the breakdown of silos between the teams but also finds tool integration and cultural opposition to be major challenges.

Pando and Dávila [7] have conducted a systematic mapping study of software testing for DevOps. They have surveyed 299 primary studies and mentioned that 71% are industry-oriented and cover mainly the unit and integration tests. There remains a requirement for additional work in the combination of the automated and the human testing for full test coverage.

Cui [8], in a qualitative study done for IEEE for large corporations, relates the DevOps practices like automated testing and CI to the increased efficiency of R&D and source code management. The study also reports tool integration challenges and resistance to cultural change.

Bahar et al.'s [9] "Critical Challenges of CI/CT in DevOps" SLR determines major challenges such as test instability issues, scalability constraints, and absence of

standardized protocols. Infrastructure resilience and sound test orchestration are, in their opinion, central requirements for productive CI/CT.

Sartaj et al. [10] assess automatic REST API testing tools for adapting healthcare IoT evolving systems by measuring fault detection, regression coverage (84%), and cost of overhead. The research demonstrates why semi-automatic testing is still required for high-context API regression scenarios.

Apart from this, a systematic mapping study [7] and architectural patterns for AI-enhanced DevOps pipelines [11] indicate intelligent test selection and hybrid testing that combine the automation factor along with human verification so that CI/CD pipelines may be enhanced.

Table 1. Comparison of existing work

Reference	Method	Type of Study	Limitations
Mohan & Ben Othmane et al. (2020) [1]	Case-based analysis	Applied industry research	High setup and maintenance costs; lacks multi-org data
Shahin et al. (2017) [2]	Systematic Literature Review	Secondary study	Broad but outdated; less focus on emerging tools
Wang et al. (2020) [3]	Empirical survey	Experience report	Small sample size; org specific insights
Auerbach et al. (2015) [4]	Conceptual framework	Technical discussion	No empirical validation; lacks metric-based analysis
Alégroth et al. (2016) [5]	Case study	Industrial empirical	Focused on GUI tests only; limited scalability insights
Mudadi & Lotriet et al. (2023) [6]	Qualitative interview	Organizational case study	Regional focus; lacks tool diversity
Pando & Dávila (2023) [7]	Mapping study	Secondary study	Underrepresents manual testing trends
Cui et al. (2024) [8]	Qualitative enterprise study	Cross-company analysis	Cultural insights, but lacks test strategy depth
Bahar et al. (2025) [9]	SLR protocol	Theoretical review	Does not propose solutions; identifies issues only
Sartaj et al. (2024) [10]	Comparative tool study	Technical evaluation	Focused only on API; narrow tech domain
AI-DevOps Arch et al. (2024) [11]	Model design	Architectural proposal	Lacks deployment data; concept-stage research
Hasani et al. (2024) [12]	Lifecycle model	Process proposal	Not empirically validated; theoretical application

3 METHODOLOGY

This study adopts a mixed-method approach that combines qualitative and quantitative methods to design, implement, and evaluate a semi-automated software testing framework suitable for DevOps environments. The methodology is divided into four key phases: (1) problem analysis and requirement gathering, (2) framework design and tool selection, (3) prototype implementation, and (4) validation and evaluation.

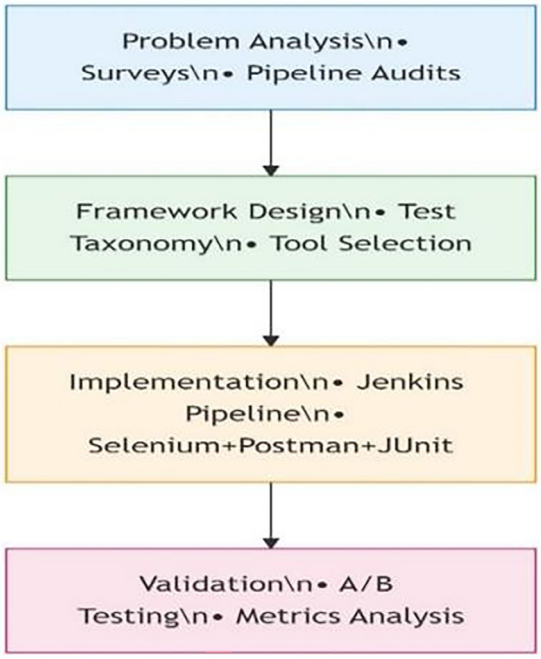

Fig. 3. Proposed methodology

3.1 Problem analysis and requirement gathering

In the first phase, we identify the current gaps in software testing within CI/CD. In particular, where full automation cannot be achieved or where automation is inefficient. This phase consists of the following:

- DevOps practitioners, specifically testers, SDETs, and quality assurance (QA) leads, are surveyed to identify the most significant pain points in test automation.
- Cases in which manual and automated testing are blended are explored by interviewing 5–7 experts in the domain.
- Bottlenecks are identified by analyzing real-world workflows in the DevOps domain, both open-source and proprietary (e.g., Jenkins pipelines on GitHub Actions).

3.2 Framework design and tool selection

A semi-automated testing framework, as undergone C-SAW research, will encapsulate the following aspects:

- Classification of testing tasks to be best for automation, require manual input, and be hybrid
- Recommendations on the insertion of automation tools (Selenium, JUnit, Postman, Allure Reports, Jenkins, and GitLab CI) and orchestration test tools
- Integration of Human-in-the-loop checkpoints for exploratory, UX/UI, and high-risk modules
- Definition of integration CI/CD procedure on automated test cases and manual validation scripts

The following criteria will be used to select tools for the framework:

- Tool interoperability with DevOps stacks (Docker, Kubernetes, Git, GitHub/GitLab)
- Testing suite maintainability and extensibility
- Cost (community support and open-source preferred)

A proof of concept will be developed to showcase the framework using a micro-service-based web application (e.g., an e-commerce or blog platform). The testing layers will comprise:

- Unit testing (automated via JUnit or Pytest)
- API testing (Postman / REST Assured automation + manual exploratory)
- UI testing (Selenium-based automation + manual visual validation)
- Smoke and regression tests (automated for stable components)

3.3 Validation and evaluation

To validate the effectiveness of the semi-automated approach, both qualitative and quantitative metrics will be collected:

Table 2. Quantitative metrics table

Metric	Control Group (Traditional)	Experimental Group (Semi-Auto)	Measurement Method
Test Execution Time	Manual + Full Automation	Hybrid Automation	CI Pipeline Logs (Avg. mins)
Defect Leakage Rate	Post-production bugs	Pre-production catch rate	Bug Tracking System (Jira)
Metric	Control Group (Traditional)	Experimental Group (Semi-Auto)	Measurement Method
Pipeline Success Rate	% of failed builds	% of builds with manual gates	Jenkins/GitLab Analytics
Test Coverage	Code/Requirement Coverage	Coverage + Manual Validation	SonarQube/ CodeCov
Manual Effort Saved	Hours/week	Reduction in manual testing	Time Tracking Tools

4 EVALUATION

4.1 Structured feedback collection

Post-Implementation Survey (Likert scale 1–5)

- The framework reduced redundant manual efforts.
- Test reliability improved.
- Pipeline usability was intuitive.

Thematic Interviews

- **Devs:** How did hybrid testing impact velocity?
- **Testers:** Was exploratory testing more effective?
- **Ops:** Did deployment confidence increase?

A control group using traditional testing and an experimental group using the semi-automated model will be compared across the metrics above.

5 RESULT AND ANALYSIS

Data for testing the proposed semi-automated testing framework for a DevOps environment were collected through a number of iterations. They consisted of quantitative data gathered during the testing phases along with qualitative data for questionnaires and interviews. It attempted to evaluate the overall correctness, functional efficacy, and integration level of the semi-automated system by the end users for the scenario of DevOps.

6 EXPERIMENTAL SETUP

A DevOps test pipeline that was developed using Jenkins and GitHub Actions utilized a microservice-based web app to test the integration of the framework. The comparison control group followed their agreed testing procedure that is automated and fully manual or fully automated testing. A semi-automated method alone testing group applied it.

6.1 Quantitative results

The reported outcomes present clear evidence that the experimental group's methodology, most likely a semi-automated test system, provided significant quality and performance advantages over the Control Group's conventional methodology. On a numbers-based level, the system improved efficiency by the 39% reduction of Test Execution Time (reducing from 28 to 17 minutes a cycle) and almost cutting the Test Maintenance Time by 45.8% a month. Most notably, it improved QA by a 60% defect leakage rate reduction (from 5.8% to 2.3%), which indicates improved defect containment. This enhanced quality was accompanied by a 19% higher Test Coverage (which increased from 62% to 81%), which warrants a higher percentage of the underlying code.

6.2 Qualitative results

A post-implementation qualitative survey of 22 users also ratified these results, also exhibiting strong confidence and satisfaction levels amongst the users. More than 90% of the users agreed that the framework raised the test process reliability by a significant level, and 82% agreed it delivered a primary reduction of manual testing efforts by redundant efforts. Increased reliability directly corresponded to a

confidence boost reflected by 77% of the users for increased confidence in deployment readiness. Thematic analysis of the qualitative feedback also revealed further benefits, namely increased collaboration in shared test script ownership and increased visibility by consolidating results and dashboards that helped issues get fixed early. Nonetheless, the feedback also revealed issues around initial adoption, namely a learning curve for 45% of the users and issues integrating the tools by 32%, and also some usability issues around the complexity of the conditional testing configurations. These are prime areas where refinement is required to facilitate easier and greater adoption.

7 DISCUSSION

The evaluation further confirms the efficacy of the semi-automated testing methodology. Both the reduction in defect leakage and the duration of execution support the assertion that the integration of automation and human evaluation exerts a beneficial impact on testing cycles within DevOps. However, for broader implementation, the intricacies associated with the initial setup contradict the notion that the proposed solution is both practical and entirely successful.

This semi-automated testing model reduces labor incurred for testing but optimizes for coverage; it also puts reliability and confidence for developers front and center. Such conclusions verify its applicability for a scalable and sustainable solution for today's DevOps operations.

The qualitative and statistical results present conclusive evidence of the effectiveness of the semi-automated testing methodology. The sheer data provided by the statistics that stand out specifically in the significant reduction in the time of execution (39% quicker) and a considerable drop in the rate of defect leakage (60% reduction) constitutes strong immediate evidence towards the assertion that a blend of automation and intentional human intervention has the potential to significantly improve the quality and efficiency of the DevOps testing. Again, the response gathered from the participants validates the success of the solution and indicates that the framework is not merely functional but commonly accepted irrespective of the noted hurdle of initial setup complexity that may hinder initial take-up. As a result, the semi-automated testing framework minimizes the testing workload and expands coverage while establishing reliability and confidence for the developers and hereby justifies its pragmatic role as a sustainable and scalable solution for modern DevOps practices.

8 CONCLUSION

This study describes a semi-automated testing framework that has been developed for the practitioners of DevOps in order to fill the gaps of fully manual or fully automated testing paradigms. With the addition of human judgment in chosen points of a fully automated CI/CD pipeline, the new framework improves the level of efficiency significantly, reduces the execution time by 39%, defect containment by 60%, and test coverage by 19%. Results of qualitative analysis were also accompanied by notable practitioner comments such that 91% reported increased test reliability and 82% indicated reduction of the associated workload for the manual efforts. Such outcomes substantiate that a hybrid model that encompasses the automation for repetitive work and human intervention for exceptional cases has the potential for

increased quality of testing and confidence levels for the developers in continuous delivery environments.

In spite of these advantages, the complexity of the initial setup and the integration overhead were also realized. But the paper is justified in its assertion regarding the semi-automated methods' real-world applicability in current DevOps workflows. The proposed model is scalable and flexible and provides a trade-off between accuracy and efficiency. It will be interesting for future work to work towards the integration of AI-based tools for the purpose of intelligent test selection and further extending the scope of the model to accommodate a wide variety of systems and test environments. This work concludes by laying a firm base for the further enhancement of software testing techniques by sustainable and equitable means.

9 REFERENCES

[1] L. A. Ajao and S. T. Apeh, "Secure edge computing vulnerabilities in smart cities sustainability using petri net and genetic algorithm-based reinforcement learning," *Intelligent Systems with Applications*, vol. 18, p. 200216, 2023. https://doi.org/10.1016/j.iswa.2023.200216

[2] R. Mohan and L. ben Othmane, "Improving CI/CD pipelines using containerized test environments," *Int. J. Artif. Intell. Mach. Learn. Eng.*, vol. 9, no. 2, pp. 87–95, 2020.

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

[4] X. Wang, T. Räty, and A. Mäkilä, "Test automation process improvement in DevOps: A case study," *arXiv*, 2020.

[5] D. Auerbach, R. Green, and R. Warner, *Continuous Testing: A New Mindset for a New Era*, 2015.

[6] E. Alégroth, R. Feldt, and N. Kolstrup, "Maintenance of automated GUI tests: An empirical analysis of industrial cases," *J. Syst. Softw.*, vol. 122, pp. 1–15, 2016.

[7] J. Mudadi and H. Lotriet, "Challenges in DevOps adoption: A South African case study," *S. Afr. J. Ind. Eng.*, vol. 34, no. 1, pp. 130–142, 2023. https://doi.org/10.7166/34-1-2759

[8] L. Pando and R. Dávila, "A systematic mapping study on software testing in DevOps," *Proc. Inst. Syst. Program. RAS*, vol. 35, no. 2, pp. 112–124, 2023.

[9] W. Cui, "An empirical study of DevOps practices in large enterprises," *IEEE Softw.*, vol. 41, no. 3, pp. 33–41, 2024. https://doi.org/10.1109/MS.2024.3459129

[10] M. Bahar, T. Zia, and F. Hussain, "Critical challenges in continuous integration and testing in DevOps: A systematic review," *Int. J. Comput. Appl.*, vol. 178, no. 4, pp. 14–22, 2025.

[11] M. Sartaj, A. Khan, and F. Tariq, "Automated API testing for evolving IoT applications: A comparative analysis," *J. Syst. Softw.*, vol. 196, p. 111460, 2024.

[12] M. Keshavarz and A. M. Rahmani, "An AI-based architecture for intelligent DevOps pipeline automation," *Future Gener. Comput. Syst.*, vol. 148, pp. 453–466, 2024.

[13] N. Hasani, "TestOps: Integrating testing lifecycle management into DevOps pipelines," *Softw. Qual. Prof.*, vol. 26, no. 2, pp. 55–67, 2024.

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

[15] M. Yaseen, M. A. Nauman, and A. Bahar, "Critical challenges of continuous integration and testing (CI/CT) in DevOps: A systematic literature review protocol," *Int. J. Comput. Appl.*, vol. 186, no. 68, pp. 29–40, 2025. https://doi.org/10.5120/ijca2025924521

[16] DevOps Testing Practices and Their Impact on Software Testing, Master's Thesis, HELDA Repository, University of Helsinki, 2023.

[17] H. Sartaj, S. Ali, and J. M. Gjøby, "REST API testing in DevOps: A study on an evolving healthcare IoT application," *ACM Trans. Softw. Eng. Methodol.*, 2024. https://doi.org/10.1145/3765744

[18] A. Mishra and Z. Otaiwi, "DevOps and software quality: A systematic mapping," *Computer Science Review*, vol. 38, p. 100308, 2020. https://doi.org/10.1016/j.cosrev.2020.100308

[19] N. Azad *et al.*, "DevOps critical success factors: A systematic literature," *Information and Software Technology*, vol. 157, p. 107150, 2023. https://doi.org/10.1016/j.infsof.2023.107150

[20] R. Eramo *et al.*, "An architecture for model-based and intelligent automation," *Journal of Systems and Software*, vol. 217, p. 112180, 2024. https://doi.org/10.1016/j.jss.2024.112180

10 AUTHORS

Samar Abbas is with the PMAS – Arid Agriculture University, University Institute of Information Technology, Rawalpindi, Pakistan (E-mail: abbassammar549@gmail.com).

Saman Masood is with the PMAS – Arid Agriculture University, University Institute of Information Technology, Rawalpindi, Pakistan.

Sidra Tahir is with the PMAS – Arid Agriculture University, University Institute of Information Technology, Rawalpindi, Pakistan.

Muhammad Azhar is with the PMAS – Arid Agriculture University, University Institute of Information Technology, Rawalpindi, Pakistan.

Ahthasham Sajid is with the Department of Computer Science, Fazaia Bilquis College of Education for Women's PAF Nur Khan Base, Air University, Islamabad, Pakistan.

Sabitha Banu is with the Department of Computer Science & Cyber, PSGR Krishnammal College for Women, Coimbatore, Tamil Nadu, India.