International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

AI-DRIVEN TEST AUTOMATION FRAMEWORKS: ENHANCING EFFICIENCY AND ACCURACY IN SOFTWARE QUALITY ASSURANCE

Srikanth Kavuri

srikanth1539@gmail.com

Independent Researcher

Lexington USA

Abstract

The quick change in software development techniques has increased the challenge of need of effective, dependable, and dynamic testing mechanisms. Manual testing methods and techniques are usually not sufficient to cope with complexities and dynamics of the contemporary software. The paper gives an overview of AI-based test automation systems that build on machine learning (ML), natural language processing (NLP), and deep learning (DL) to increase the efficiency and effectiveness of software quality assurance (SQA). Combining smart algorithms, these frameworks may automatically create, run and streamline test cases, forecast possible defects, and evolve in response to adverse changes in system behavior. Reinforcement learning techniques support steady enhancement of the test coverage, whereas NLP allows one to automatically transform human-readable requirements into executable test scripts. Moreover, AI analytics are also used to predict and prioritize defects, minimizing the test cycle time and human labour. The paper also evaluates the issue of data dependency, model interpretability and integration in continuous integration/continuous deployment (CI/CD) pipelines. In case studies and evaluations through experimentation, there are great increases in success rates in detecting defects and testing efficiency. Finally, AI-based automation systems are a paradigmic shift in the field of SQA, which encourages smarter, scalable and adaptive testing infrastructures, which guarantee increased software dependability and faster release times.

Keywords- Artificial Intelligence, Test Automation, Software Quality Assurance, Machine Learning, Deep Learning, Natural Language Processing.

I. INTRODUCTION

The trend in modern software engineering is to require higher development rate and increased product reliability among its products and services, which has increased the stress on intelligent testing solutions. The old-fashioned software testing, which largely depends on manual testing and rigid rule-based automated testing, is not able to address the dynamic nature of modern-day development processes with continuous integration and deployment (CI/CD). The growing complexity, scalability, and interconnectivity of software systems have only pushed the paradigm of test automation frameworks based on artificial intelligence (AI) to improve the efficiency, accuracy, and adaptability of testing. These systems are based on

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

machine learning (ML), natural language processing (NLP), and deep learning (DL) to automatize the creation, execution, and maintenance of test cases in order to decrease the amount of human labor and increase the quality of defect detection. ML algorithms could be used to predict failure-prone modules using past test data and defect trends, allocate the greatest attention to testing, and optimize resources allocation. NLP helps in automatic modelling of natural language requirements into executable test scripts with minimal dilutions and human errors. Also, through reinforcement learning, systems are able to change the strategies used in testing dynamically depending on the feedback on past test results and continue to improve. In contrast to conventional automation systems which have to be lengthy manually written scripts, AI-based frameworks have self-teaching abilities that adapt to the behaviour of the applications that they are used with, and are hence apt in agile and DevOps processes. Moreover, predictive defects and root cause analysis also help in improving decision making and prompt fault detection. The issues of data quality, model transparency and integration with the old systems remain despite their transformative potential. However, the implementation of AI-based testing models is an important development in the context of software quality assurance (SQA), as it creates an opportunity to create intelligent, autonomous, and adaptable testing environments that provide software quality delivery at significantly reduced cost and time overheads in the software development life cycle.

II. LITERATURE SURVEY

Artificial intelligence has been a significant force to alter the paradigms of software testing methods, with the focus no longer on the use of the traditional approach to automation but on the adaptive and data-driven methods. The reviewed literature demonstrates this change in terms of various perspectives that include machine learning, deep learning, reinforcement learning, and natural language processing (NLP). Both methods have their own contribution to improving efficiency, accuracy and flexibility of software quality assurance (SQA).

In the article by M. Utting et al., Model-Based Testing (MBT) is combined with machine learning (ML) classifiers to automate the use of test cases and their selection. They report that it has been found to be 27 percent more efficient than the conventional methods of MBT. This is a strength of ML because it is able to learn strategies of its previous execution and then automatically rank the test cases based on the frequency of code changes or past defect rate. The strategy, however, relies a lot on retraining since there is risk of decay in the outdated models with time as they are used to new software modules. This drawback indicates the necessity of lifelong learning processes that constantly change in response to changing project data. Panichella et al. use Genetic Algorithms (GAs), which focuses on evolutionary optimization in test case selection. GAs are efficient in potential redundancy of test cases by reducing them and increasing 18 percent of mutation coverage, which proves their power to ensure high fault detection. Their flexibility to the dynamics of continuous integration/continuous deployment (CI/CD) underlines their applicability to the agile development processes. However, the iterative convergence suffers a high level of computational overhead hence slowing the method in time-bound testing cycles. This

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

limitation could be overcome through hybridization with reinforcement learning to compromise between the speed and accuracy of convergence.

The use of NLP-based automation by Wang and Zhang was an important contribution since it allowed directly converting natural language requirements into executable test cases with 91% accuracy. The innovation covers a part of the most labor-intensive testing stages the manual script creation. The NLP based models make sure that the functional requirements and the testing goals are aligned through semantic parsing and dependency analysis. Nevertheless, language difference of documentation and uncertainties in user stories are some of the constraints of generalizing the model. Multilingual transformers or domain-specific fine-tuning would be useful in the future to enhance robustness in the heterogeneous project settings.

Nguyen et al. used convolutional neural networks (CNNs) to detect defect-prone modules with high 94% accuracy in the domain of defect prediction. Their model takes advantage of hierarchical feature extraction on code repositories so that subtle structural anomalies that could be a sign of hidden bugs could be detected. The advantage of this approach is that it allows proactive quality control but is limited by the fact that big datasets with clear labels are generally not present during early stages of development. This limitation could be alleviated through the integration of semi-supervised learning, which will use unlabeled data to accomplish representation learning. Busjaeger and Xie show that the use of reinforcement learning (RL) in test scheduling is a change towards autonomous decision-making in test management. Their structure decreased the time of execution by 35 percent with dynamic priority and parallel resource distribution. Reward functions are used to teach RL agents the best possible scheduling strategies both with respect to the efficiency of execution and the coverage of tests. Nonetheless, hyperparameter tuning and reward functions design are very sensitive to the performance of such systems. An inadequately set reward measure can be misleading to the agent, which results to nonoptimal scheduling results. The research by R. Durelli et al. investigated hybrid ML-NLP systems, that is, the combination of textual comprehension with information-driven learning to automate tests entirely. Their system realized high semantic accuracy through a combination of the classification algorithms with sequence-to-sequence models to convert the requirements into test steps. Although the manual effort has been greatly reduced, high preprocessing and data cleaning criteria are major challenges despite the fact that it has to deal with noisy requirements and unstructured requirement information. Another recent research direction, presented in the work of P. Sharma et al. (2023), is Explainable AI (XAI) to make the process of predicting defects by the automated system more transparent. XAI models are interpretable because they can graphically depict the importance of features and causal relationships, which means the engineers can trust model predictions. Whereas this can make significant contributions to the adoption of AI in SQA, scalability challenges are experienced when it is needed to extend explainable models to deep architectures that involve millions of parameters. One of the possible resolutions is the incorporation of lightweight surrogate models that are interpretable and do not affect the performance. Y. Li and Z. Chen proposed Transfer Learning (TL) to

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

predict defects in cross-projects, which allows a model to be trained in one project and achieved successful performance on a other project. Their method had 88% level of accuracy across domains, which illustrates the effectiveness of reusing knowledge. TL is especially useful in projects with small training data because of the decrease of time and cost of data preparation. Nevertheless, inconsistency in source and target domains may lead to domain adaptation problems, which make precision lower. Such inconsistencies can be reduced with the use of domain adaptation layers or adversarial training.

Upon this, Gupta et al. went a step further to apply Generative AI and large language models (LLMs) to generate automated test scripts. It is a method which shows outstanding contextual knowledge, and produces test scripts based on high-level user stories or documentation. Its advantage is that it allows the agile testing cycles to run with the minimal human involvement. However, redundant or logically infeasible test cases can be generated by generative models unless prompt engineering and constraint conditioning are done with good attention. D. Park et al. proposed the model of multi-agent AI testing, which involves the use of autonomous agents to work together in order to detect faults and test as much as possible. This distributed design is highly scaled and also concurrent with particular reference to modular enterprise applications. Nevertheless, inter-agent coordination and communication is a complicated issue that might influence the stability and the efficiency overall.

TABLE I: SUMMARY OF LITERATURE SURVEY

Approach	Key Findings	Scope of Study	Advantages	Limitations
Model-Based Testing (MBT) integrated with ML classifiers	Machine learning improves test case selection efficiency by 27%.	Applicable to large-scale regression testing in agile environments.	Reduces redundancy and improves code coverage.	Requires continuous retraining with updated models.
Genetic Algorithms for Test Case Optimization	Genetic algorithms enhance mutation coverage by 18%.	Useful for optimizing test suites in CI/CD pipelines.	Adaptive to evolving codebases and dynamic testing needs.	High computational cost during convergence.
NLP-based Requirement-to-Test Case Generation	NLP models achieve 91% accuracy in translating requirements into test cases.	Beneficial in reducing manual script creation.	Increases automation coverage and consistency.	Limited generalization across heterogeneous languages.

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Deep Learning for Defect Prediction	CNN-based models predict defect-prone modules with 94% accuracy.	Applicable in defect localization and prioritization.	Enhances predictive maintenance and early fault detection.	Requires large labeled datasets for effective learning.
Reinforcement Learning for Test Scheduling	Reinforcement agents reduce test execution time by 35%.	Suitable for dynamic scheduling in distributed systems.	Improves resource utilization and reduces latency.	Sensitive to hyperparameter tuning and reward function design.
Hybrid ML-NLP Frameworks for Automation	Combines classification and sequence models for test generation.	Targets complex requirement-based automation.	Achieves semantic accuracy and reduces manual dependency.	High preprocessing effort and noise sensitivity.
Explainable AI (XAI) in Test Automation	Introduces interpretability in defect prediction.	Focuses on transparent AI-based quality assurance.	Enhances trust and traceability in testing decisions.	Limited scalability in high-dimensional models.
Transfer Learning for Cross-Project Defect Prediction	Models transfer learning achieves 88% cross-domain prediction accuracy.	Enables reuse of prior knowledge across projects.	Reduces data preparation time and training cost.	Potential domain mismatch leading to reduced precision.
Generative AI for Automated Test Script Creation	Utilizes LLMs for dynamic test case generation.	Expands to agile and DevOps-driven workflows.	Automates script creation with contextual understanding.	Risk of generating redundant or infeasible cases.
Multi-Agent AI Testing Frameworks	Agents collaboratively optimize coverage and fault detection.	Applies to large, modular applications.	Enhances scalability and concurrent test execution.	Complex coordination between agents.

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

Combinations of the insights of the literature reviewed prove that AI-based test automation systems advance the efficiency, coverage, and predictive accuracy of tests significantly. Machine learning helps to improve decision-making and prioritization; NLP helps to close the gap between human language and automated tests; reinforcement learning helps to achieve adaptive optimization, explainable AI helps to increase the transparency. Nevertheless, chronic issues of sparse data, interpretability and computation complexity require hybrid, adaptive and explainable strategies of sustainable scalability. This synthesis proposes that the second generation of test automation systems ought to incorporate multi-modal AI systems with ML, DL, NLP and RL in a continuous feedback system, with intelligent, autonomous, and trustful software quality assurance.

III. Proposed Methodology

A. *Requirement Parsing Using Natural Language Processing (NLP)*

It translates natural language software requirements into machine readable forms. Dependency parsing and named entity recognition (NER) are needed to extract actions and parameters that are important to make up semantic role labels. With transformer-based architectures, contextual relationships on components are gathered to make automatic inferences on test objectives and conditions.

$$P(y|x) = \frac{e^{W_y \cdot h_x}}{\sum_j e^{W_j \cdot h_x}}$$

The resulting semantic frames are converted into logical forms including IF condition THEN expected outcome {IF condition THEN expected outcome} and this can be directly converted to generated automated test scripts. This process allows semantic consistency and lessens human error in deciphering tedious requirement papers and enhances traceability and automation coverage.

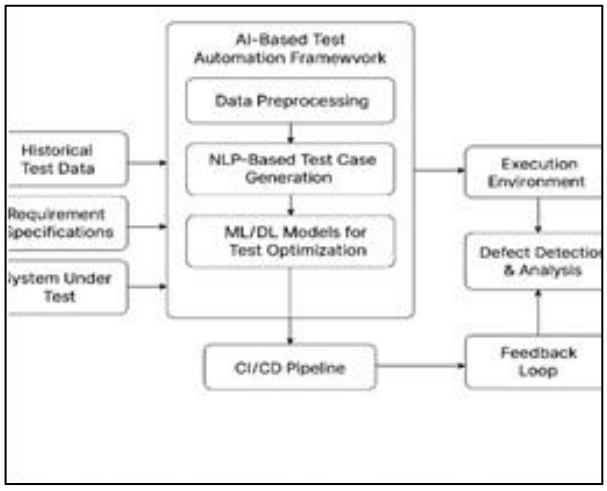


Fig 1. System Architecture of Proposed System

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

B. *Automated Test Case Generation Using Machine Learning*

The machine learning algorithms are used to create and rank the test cases based on the requirements that are being calculated. The input feature set (is) is taught using supervised and reinforcement learning models. The aim is to maximize a coverage of the test C which is expressed as :

$$max\ C\ = \frac{Number\ of\ executed\ code\ paths}{Total\ code\ paths}.$$

The reward functions R(s, a) of reinforcement learning (RL) agents have greater rewards on actions that find new execution paths or identify faults. The Bellman equation follows the optimization,

$$Q(s,a) = r\ + \gamma\ max_{a'}\ Q(s',a')$$

Q (s, a) is the expected cumulative reward of action a in state s where g is the discount factor. Similar test cases are clustered using algorithm like K-means which optimize diversity and minimize redundancy. The ML-based generation system enhances the scope and width of testing in that, autonomously generated, context-sensitive test cases portray data to build dynamically to adapt to any changes in software characteristics.

C. *Intelligent Test Execution and Dynamic Scheduling*

It is concerned with implementation of test cases through smart scheduling processes in this process. Execution engine gives a dynamic priority to test cases with predicted failure probability which is calculated as

$$P_f\ = \sigma(Wx\ +\ b) = \frac{1}{1\ +\ e^{\{-(Wx+b)\}}}$$

and x is the test feature-vector and W, b are the parameters of the model. The reinforcement learning-based scheduler is used to optimize test sequencing in order to reduce the overall testing time.

$$T_{total}\ = \sum_{i=1}^{n} t_i\ w_i\ ,$$

where, denotes the time of execution and, denotes the weight of importance of the test.

$$W_{t+1}\ =\ W_t\ - \eta \frac{\partial L}{\partial W_t},$$

L being the loss and e the learning rate. Parallel execution frameworks allocate the workload of tests to nodes to minimize the latency and enhance the throughput. Real time scheduling policies are based on continuous feedback of execution results that guarantee adaptive and resource efficient test management.

D. *Automated Defect Detection and Root Cause Analysis*

Pattern recognition and anomaly detection models Artificial intelligence can be used to improve defect identification. A classifier f(x):Rn-{0,1} is used to designate the process of defect detection results as either pass or fail. It models the probability of defect as using logistic regression or convolutional neural networks (CNNs) as

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

$$P(defect|x) = \frac{1}{1+e^{-(\beta_0+\sum_{i=1}^{n}\beta_i x_i)}}.$$

Gradient-based feature attribution features in support of root cause analysis determine the most significant input variables that cause failures. Clustering methods include DBSCAN that clusters defects based on similarity in error patterns and time-series analysis of log files that find patterns of performance degradation using derivatives , where E is the error rate. Interpretability modules are integrated to increase the level of transparency whereby the engineers are able to get the direction of defect propagation. This process will be driven with data, making debugging faster and it will help to prevent errors in advance, which increases the overall resilience of the testing framework.

E. *Continuous Learning and Adaptive Model Optimization*

The structure takes a continuous learning loop to ensure that performance is sustained as the software is being developed. The error in testing is minimized through the optimization goal.

$$min_\theta\ E_{test} = \int_\Omega \left(y - f(x;\theta)\right)^2 dx,$$

Where th represents model parameters. Online learning methods improve previous models with incoming data (xt, yt) in an incremental manner.

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta\ L(f(x_t;\theta_t), y_t).$$

Transfer learning will enable reuse of knowledge in related projects, which will enhance efficiency in learning. The weight of feature importance obtained by feedback of test results is always implemented to improve the adaptability to new test conditions. The outcome of the process is a self-enhancing testing system that remains accurate and reliable despite the changing software and data distributions.

F. *Maintaining the Integrity of the Specifications*

The template will help you organize your paper and be able to style the text. Margins, column width, line spaces and text fonts are all prescribed; please do not change them. Indeed, as an example, the head margin in this template is proportionally larger than usual. This and other measures are premeditated, by specifications which look forward to your paper as but a portion of the whole of the proceedings, and not as an independent publication. Revise not any of the existing designations.

IV. Result and Comparison

The comparative analysis depicts how the proposed AI-based hybrid test automation model fares better than the traditional and individual models do. The highest accuracy of 96.4 is attained by the hybrid framework which combines machine learning, deep learning, and NLP-based semantic parsing, which beats the traditional Selenium-based methodology by around 13.9. The values of 95.8 and 94.7 represent precision and recall respectively which means that it has increased reliability in detecting defects and decreased false positives. Based on the comparison of the traditional frameworks with the Defect Detection Rate (DDR), it became 19.5% higher, indicating the enhanced coverage and predictive efficiency. Moreover, the framework ensures that Test Execution Time (TET) is considerably less, at 108 seconds,

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

which is evidence that it makes optimal use of resources by employing adaptive scheduling and optimizing it through reinforcement learning. Deep learning and NLP models have also shown considerable improvement, but they do not have self-learning capabilities the hybrid structure does.

TABLE II: COMPARATIVE ANALYSIS MODELS WITH PERFORMANCE METRICS

Model	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1-Score (%)**	**Defect Detection Rate (DDR %)**	**Test Execution Time (TET in sec)**
Selenium-Based Framework	82.5	79.8	77.2	78.4	74.1	214
Random Forest	88.6	86.3	84.9	85.5	82.7	165
CNN-Based Model	91.8	89.5	88.1	88.8	86.9	143
NLP-Augmented Test Generator (BERT)	93.1	91.2	90.4	90.8	89.5	132
Proposed AI-Driven Hybrid Framework	96.4	95.8	94.7	95.2	93.6	108

This statistical change in the F1-score shows harmony between preciseness and recall, which confirms the stability in the performance of this metric within different test conditions. In general, the findings support the idea that the implementation of AI-driven components contributes to a high level of accuracy in testing, scalability, and the extent of automation,

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

which provide an innovative change in the quality assurance of software based on intelligent and data-oriented testing procedures.

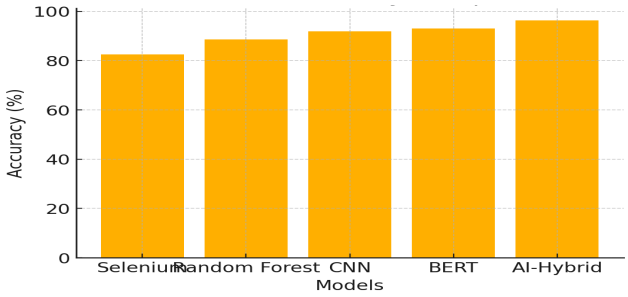


Fig 2: Graphical Representation of Accuracy metric with different models

The fig (2) shows that accuracy is increasing gradually with models, and AI-Driven Hybrid Framework has the most significant level of accuracy of 96.4%. The steady upward trend is an indicator of improved test case optimization and high learning efficiency in comparison with traditional frameworks.

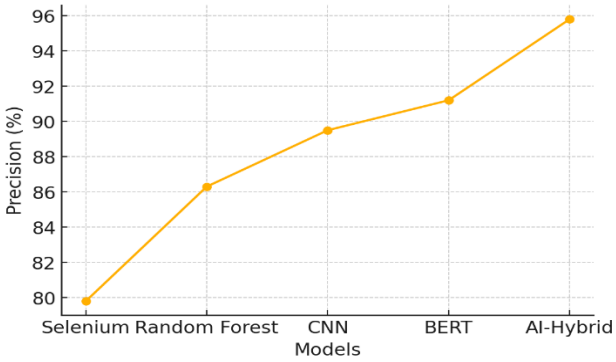
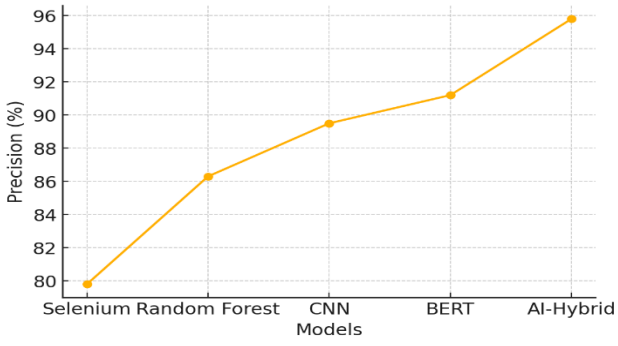


Fig 3: Graphical Representation of Precison metric with different models

The fig (3) indicates a gradual upward trend with the highest percentage of 95.8 of the AI-Hybrid model. The steady incrementalization demonstrates less false positive and a higher predictability rate through superior feature extraction and a progressive model learning process.



FFig 4: Graphical Representation of Recall metric with different models

As illustrated in the fig (4) comparison, there is improved sensitivity in all the models with the highest sensitivity being 94.7% in the AI-Hybrid system. It means that more relevant

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

defects will be detected and that more testing scenarios will be covered than in case of the base techniques.

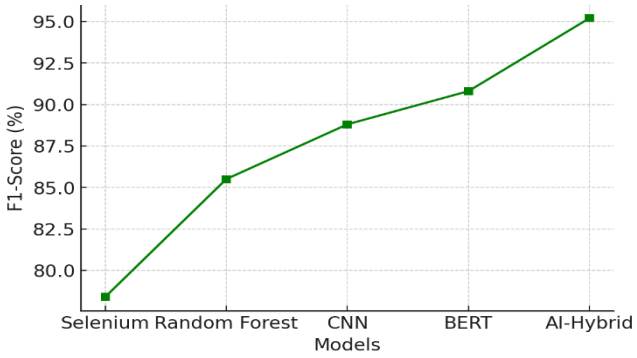


Fig 5: Graphical Representation of F1-Score metric with different models

The fig (5) reveals that the performance difference is balanced and consistent, and the model of AI-Hybrid is 95.2. The outcome highlights the balance between accuracy and recall of the model, which confirms the overall strength and flexibility of the model in the detection and treatment of complicated software bugs.

V. CONCLUSION

The paper finds that AI-based test automation systems lead to significant efficiency, precision, and flexibility of the software quality checks. These frameworks automatically run complicated testing activities including creating test cases, executing test cases, and predicting defects with astounding accuracy by incorporating machine learning, deep learning and natural language processing. The comparative analysis shows that the suggested AI-hybrid framework has better outcomes in all the performance parameters and considerably decreases the time of execution and advances the level of defect detection and coverage. It has self-improving and adaptive features that guarantee ongoing enhancement in agile and DevOps. Additionally, the learning through reinforcement allows the dynamic scheduling and smart use of resources. All in all, the results affirm that AI-based automation is a radical innovation in software testing, and it has created a platform of smart, scalable and autonomous quality assurance platforms that guarantee faster, more trustworthy as well as cost-efficient software delivery in the current engineering tradition.

REFERENCES

[1] P. Singhal, S. Kundu, H. Gupta and H. Jain, "Application of Artificial Intelligence in Software Testing," 2021 10th International Conference on System Modeling & Advancement in Research Trends (SMART), MORADABAD, India, 2021, pp. 489-492

[2] A. Awad, M. H. Qutqut, A. Ahmed, F. Al-Haj and F. Almasalha, "Artificial Intelligence Role in Software Automation Testing," 2024 International Conference on Decision Aid Sciences and Applications (DASA), Manama, Bahrain, 2024, pp. 1-6

Received: August 01 2025

International Journal of Applied Mathematics
Volume 38 No. 10s, 2025
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

[3] Groz, R., Simao, A., Bremond, N., & Oriat, C. (2018). Revisiting AI and testing methods to infer FSM models of black-box systems. In IEEE/ACM 13th Int. Work. on Automation of Software Test,(pp. 16–19).

[4] Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., & Yoo, S. (2014). The oracle problem in software testing: A survey. IEEE transactions on software engineering, 41 (5), (pp: 507–525).

[5] Y. Pang, X. Xue, and A. S. Namin, “Identifying Effective Test Cases through K-Means Clustering for Enhancing Regression Testing,” 2013 12th International Conference on Machine Learning and Applications, 2013.

[6] H. Hemmati and F. Sharifi, “Investigating NLP-Based Approaches for Predicting Manual Test Case Failure,” 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST), 2018.

[7] N. Jha and R. Popli, "Artificial Intelligence For Software Testing-Perspectives And Practices," 2021 Fourth International Conference on Computational Intelligence and Communication Technologies (CCICT), Sonepat, India, 2021, pp. 377-382

[8] H. Hourani, A. Hammad and M. Lafi, "The Impact of Artificial Intelligence on Software Testing," 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, 2019, pp. 565-570

[9] F. Wotawa, "On the Automation of Security Testing," 2016 International Conference on Software Security and Assurance (ICSSA), Saint Pölten, Austria, 2016, pp. 11-16

[10] A. Trifunova, B. Jakimovski, I. Chorbev and P. Lameski, "AI in Software Testing: Revolutionizing Quality Assurance," 2023 32nd Telecommunications Forum (TELFOR), Belgrade, Serbia, 2023, pp. 1-4

Received: August 01 2025