International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 6, June 2025)

Comparative Analysis of Manual testing and Automated testing with Artificial and Machine Learning Implementation

[1]Tapish Atreya, [2]Dr.Ritesh Yadav

[1]*M.Tech Scholar, Department of Computer Science and Engineering, RKDF-IST, SRK University, Bhopal, M.P, India*
[2]*Associate Professor, Department of Computer Science and Engineering, RKDF-IST, SRK University, Bhopal, M.P, India*

Abstract— Artificial Intelligence (AI) and Machine Learning (ML) have brought profound changes across numerous sectors, with software development being no exception. Within the software development lifecycle (SDLC), software testing plays a vital role in validating the quality and reliability of software products. Historically, this testing phase has relied heavily on manual labor, making it time-consuming and resource-intensive. However, the rise of AI and ML has revolutionized this process by enabling automation and smarter decision-making. By leveraging AI and ML, software testing becomes more efficient and effective. These technologies facilitate the automation of complex activities such as generating test cases, executing tests, and analyzing outcomes. This automation not only accelerates the testing timeline but also enhances the precision of defect identification, thereby contributing to the delivery of higher-quality software. This paper explores the integration of AI and ML in software testing through a thorough review of existing research, an evaluation of current tools and methodologies, and the presentation of case studies that highlight tangible benefits. The literature review offers a broad perspective on recent advancements in AI and ML applications within software testing, emphasizing key techniques and findings from various studies. Furthermore, the assessment of contemporary tools highlights the functionalities of prominent AI-powered testing solutions such as Eggplant AI, Test.ai, Selenium, Appvance, Applitools Eyes, Katalon Studio, and Tricentis Tosca—each known for their unique strengths and capabilities. Included case studies provide practical examples of how AI and ML have been applied across different industries, demonstrating notable improvements in testing speed, accuracy, and overall software quality. These examples emphasize the adaptability and effectiveness of AI-enhanced testing tools across diverse environments. Additionally, the paper addresses challenges encountered when incorporating AI and ML into software testing, including concerns related to data integrity, algorithmic complexity, and ethical implications.

Keywords— Artificial Intelligence (AI), Machine Learning (ML), Software Automation, Cybersecurity, Secure Software Development, Intelligent Testing, DevSecOps.

I. INTRODUCTION

1. The Impact of AI and ML on Software Testing: Artificial Intelligence (AI) and Machine Learning (ML) are cutting-edge technologies that have made a notable impact across a wide range of sectors, including software development. Within the software development lifecycle (SDLC), testing plays a critical role in validating both the quality and reliability of software products. Integrating AI and ML into software testing can streamline and enhance the process by automating intricate tasks, thereby reducing the overall time and effort required for comprehensive testing.Over the past ten years, advancements in AI and ML have accelerated, particularly in their application to software testing. This paper investigates how AI and ML are transforming software testing by conducting a detailed review of existing research, assessing modern tools and methodologies, and showcasing case studies that highlight the real-world advantages these technologies offer.

2. Evolution of Software Testing Practices: Software testing has undergone significant transformation over the years. In its early stages, testing was performed manually by developers or specialized testers. As software systems became more sophisticated, it became evident that more organized and systematic testing methods were necessary. This realization led to the introduction of automated testing

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 6, June 2025)

tools during the late 20th century. Despite their advantages, these early tools often lacked flexibility and required considerable manual intervention to update and maintain test scripts. The emergence of Artificial Intelligence (AI) and Machine Learning (ML) has ushered in a new era of intelligent and adaptive testing solutions, designed to address many of the limitations associated with traditional testing approaches. AI- and ML-based testing methods offer solutions to many inherent constraints of manual and early automated testing, such as scalability, maintenance, and responsiveness to changes in software systems.

- Automatically generate and optimize test cases
- Reduce maintenance overhead
- Improve test coverage and fault detection
- This evolution represents a pivotal shift from rule-based automation to data-driven, intelligent testing, enabling greater efficiency, scalability, and reliability in software quality assurance

II. LITERATURE SURVEY

2.1 The Role of AI in Software Testing

Artificial Intelligence (AI) is transforming software testing by automating repetitive tasks, predicting failures, and enabling more intuitive test generation. This section presents data and visualizations to illustrate the impact of AI-driven tools on software testing efficiency and effectiveness.

Table 1: Comparison of Manual vs. AI-Driven Testing

Aspect	Manual Testing	AI-Driven Testing (e.g., Eggplant AI, Test.ai)
Test Case Generation	Manual, time-consuming	Automated, rapid
Test Execution	Sequential, slow	Parallel, fast
Defect Detection Accuracy	Moderate	High (predictive analytics)
Adaptability to Change	Low	High (dynamic learning)
Maintenance Effort	High	Low
Requirement Analysis	Manual	Automated (NLP-based)
Involvement of Non-technical Stakeholders	Difficult	Easier (NLP allows plain language input)

By using deep learning models, Test.ai can recognize and interact with complex and dynamic user interface elements, automating the testing process without the need for traditional scripting. Its NLP features further facilitate the automatic generation of test scenarios from user stories or requirements, streamlining collaboration between development and QA teams.

Chart 1: Time Reduction in Test Case Generation and Execution

	Time spent in testing tasks	
Task	Manual testing	AI- Driven testing
Test Generation	20	5
Test Execution	30	10
Result analysis	10	3
Total	60	18

It shows the time deduction in Test Case generate and execution. In Manual testing the time is prolonged. Hence, to over the automating testing is implemented.

Table 2: Key Features of Leading AI-Driven Testing Tools

Tool	AI Techniques Used	Key Features
Eggplant AI	ML, NLP	Automated test generation, predictive analytics
Test.ai	Deep Learning, NLP	UI testing, scriptless automation
Applitools Eyes	Visual AI	Visual validation, cross-browser testing
Selenium (AI add-ons)	ML	Smart locators, self-healing scripts

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 6, June 2025)

The table presents a comparative overview of four prominent AI-powered software testing tools—Eggplant AI, Test.ai, Applitools Eyes, and Selenium with AI add-ons—highlighting the diversity in their underlying AI techniques and specialized features. Eggplant AI leverages both machine learning (ML) and natural language processing (NLP) to automate the generation of test cases and apply predictive analytics. This dual approach enables the tool to model user journeys, anticipate potential failure points based on historical data, and dynamically prioritize testing efforts for maximum coverage and efficiency. Its NLP capabilities also allow it to interpret software requirements written in plain language, making test creation more accessible to non-technical stakeholders and reducing the manual effort required for test maintenance. Test.ai, on the other hand, employs deep learning and NLP to enable scriptless UI testing. Applitools Eyes distinguishes itself through the use of Visual AI, focusing on visual validation and cross-browser testing. This tool uses advanced image comparison algorithms to detect even the smallest visual discrepancies across different browsers and devices, ensuring a consistent user experience and catching issues that might be missed by conventional functional tests. Meanwhile, Selenium—augmented with AI-powered add-ons—integrates machine learning to enhance its traditional automation capabilities. These add-ons introduce smart locators and self-healing scripts, which enable Selenium to adapt to changes in the application's UI automatically. As a result, test scripts become more resilient to frequent updates or modifications in the software, significantly reducing the maintenance burden on QA teams. Collectively, these tools demonstrate how the integration of various AI techniques—ranging from ML and deep learning to NLP and Visual

III. PROBLEM STATEMENT

1. Primary Problem Areas

1. Limited Adaptive Capability Current automation systems operate within rigid, predefined parameters and struggle to adapt to unexpected scenarios, environmental changes, or evolving requirements. This limitation results in frequent system failures, reduced reliability, and increased manual intervention needs, ultimately undermining the core value proposition of automation.

2. Inadequate Pattern Recognition and Anomaly Detection Traditional automation lacks sophisticated pattern recognition capabilities essential for identifying complex system behaviors, predicting potential failures, and detecting subtle anomalies that may indicate security threats or performance degradation. This deficiency leads to reactive rather than proactive system management approaches.

3. Scalability and Complexity Management Challenges As software systems grow in complexity and scale, traditional automation approaches become increasingly difficult to maintain, configure, and optimize. The linear scaling limitations of rule-based systems create bottlenecks that impede organizational growth and technological advancement.

4. Insufficient Context Awareness Existing automation solutions often operate in isolation without comprehensive understanding of broader system context, business objectives, or environmental factors. This limitation results in suboptimal decision-making and potential conflicts between different automated processes.

2. Integration Complexity

The incorporation of AI and ML technologies into existing automation frameworks presents significant technical hurdles, including:

- **Data Quality and Availability**: AI and ML systems require high-quality, representative datasets for training and operation, yet many organizations lack sufficient data governance practices or encounter data silos that limit algorithm effectiveness.
- **Real-time Processing Requirements**: Software automation demands real-time or near-real-time decision-making capabilities, creating computational and architectural challenges for AI/ML systems traditionally designed for batch processing environments.
- **Model Interpretability and Explainability**: The "black box" nature of many AI/ML algorithms conflicts with the transparency and auditability requirements essential for enterprise software automation, particularly in regulated industries.
- **System Integration and Interoperability**: Incorporating AI/ML capabilities into existing automation infrastructure requires addressing compatibility issues, API limitations, and architectural constraints that may not have been designed to accommodate intelligent systems.

3. Operational Challenges

Resource Optimization and Cost Management AI and ML

International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347 - 6435 (Online) Volume 14, Issue 6, June 2025)

integration introduces significant computational overhead and resource requirements that must be balanced against automation benefits. Organizations struggle to optimize resource allocation while maintaining system performance and cost-effectiveness.

Skill Gap and Knowledge Transfer The successful implementation of AI/ML-enhanced automation requires specialized expertise that spans multiple domains, including software engineering, data science, machine learning, and domain-specific knowledge. Many organizations face critical skill shortages that impede adoption and implementation efforts.

Quality Assurance and Validation Traditional software testing and validation approaches are insufficient for AI/ML-integrated systems, requiring new methodologies for ensuring system reliability, accuracy, and behavioral consistency across diverse operational scenarios

4. Business Impact

Organizations that fail to effectively integrate AI and ML into their software automation strategies risk competitive disadvantage through:

- Reduced operational efficiency and increased manual overhead
- Limited scalability and growth potential
- Increased security vulnerabilities and system failures
- Higher operational costs and resource waste
- Diminished ability to respond to market changes and customer demands

5. Innovation Barriers

The current state of AI/ML integration in software automation creates systemic barriers to innovation, including:

- Reluctance to adopt advanced automation due to implementation complexity
- Limited experimentation with intelligent automation approaches
- Reduced organizational agility and technological adaptability
- Inability to leverage data-driven insights for process optimization

IV. CONCLUSION

Analysis of Manual Testing Results: Analysis of manual testing outcomes across various project sizes reveals distinct patterns that demonstrate the contextual effectiveness of manual testing approaches. Small projects, typically characterized by dynamic and agile development environments, benefit significantly from the inherent flexibility of manual testing methodologies. These projects often require rapid adaptation to changing requirements and frequent iterations, making manual testing methods particularly valuable as they can accommodate these dynamic conditions without the overhead associated with maintaining automated test scripts. The human element in manual testing provides the necessary adaptability and creative problem-solving capabilities that align well with the exploratory nature of small-scale development efforts.

Medium-sized projects present a more complex scenario where a hybrid approach proves most beneficial. In these environments, manual testing is strategically applied in areas where human insight and intuition are crucial, particularly in exploratory testing scenarios and comprehensive functional testing where nuanced understanding of user behavior and system interactions is essential. Simultaneously, these projects begin to introduce automation for repetitive tasks that can be standardized, creating a balanced testing ecosystem that leverages the strengths of both approaches while mitigating their individual limitations.

Large projects, however, demonstrate a different pattern where manual testing becomes progressively less effective due to the increased complexity and scale inherent in these systems. The sheer volume of test cases, the complexity of system interactions, and the need for frequent regression testing make purely manual approaches impractical and inefficient. Consequently, manual testing efforts in large projects are often supplemented or entirely replaced by automated strategies to improve overall efficiency and ensure scalability of the testing process. The comprehensive analysis, utilizing both quantitative data from surveys and testing metrics alongside qualitative insights gathered through detailed interviews, reveals several key strengths of manual testing. Manual testing demonstrates exceptional effectiveness in exploratory and functional testing scenarios where human creativity and intuition play critical roles in identifying potential issues. It proves particularly valuable for identifying edge cases and unusual system behaviors that might not be anticipated in automated test scripts, as human testers can apply contextual understanding and creative thinking to uncover these scenarios. Additionally, manual testing shows superior adaptability for projects with frequently changing

requirements, as human testers can quickly adjust their approach without the need for extensive script modifications or redevelopment.

Analysis of Automation Testing Results: Automation testing demonstrates significant advantages, particularly in large-scale and complex projects where test cases require frequent repetition and consistent execution. Large projects benefit substantially from automation implementation, as it dramatically improves productivity while providing reliable and repeatable results. This advantage becomes particularly pronounced in regression testing scenarios and performance testing environments where consistent execution parameters are crucial for meaningful results. The ability to execute comprehensive test suites rapidly and repeatedly enables development teams to maintain high-quality standards while accelerating development cycles.Medium-sized projects present a more nuanced scenario regarding automation adoption. While these projects can certainly benefit from automation implementation, medium-sized organizations often face practical challenges in securing the necessary budget for initial automation setup and the ongoing investment required for maintaining automated test suites. The cost-benefit analysis for medium projects requires careful consideration of the long-term testing requirements against the upfront investment and ongoing maintenance costs associated with automation infrastructure.Small projects, particularly those characterized by highly dynamic development environments, often find that the cost and effort associated with automation implementation outweigh the potential benefits. The rapid pace of change typical in small projects can result in frequent modifications to automated test scripts, potentially negating the efficiency gains that automation is intended to provide. In these scenarios, the overhead of maintaining automation infrastructure may exceed the benefits gained from automated execution.The study reveals several key advantages of automation testing that contribute to its effectiveness in appropriate contexts. Automation significantly improves efficiency and reduces overall testing time, particularly for repetitive test cases that would otherwise require substantial manual effort. It demonstrates exceptional effectiveness in regression testing, performance testing, and multi-platform testing scenarios where consistent execution and comprehensive coverage are essential. Additionally, automation increases overall productivity through the reuse of test scripts across multiple testing cycles and projects, creating economies of scale that benefit long-term testing strategies.

References

1. Mulla, M. M., & Jayakumar, S. (2021). Artificial Intelligence in Software Testing: Applications and Challenges. Procedia Computer Science, 192, 2203-2212.
2. Amershi, S., et al. (2019). Software Engineering for Machine Learning: A Case Study. Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, 291-300.
3. Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. ACM Computing Surveys, 41(3), 1–58. https://doi.org/10.1145/1541880.1541882
4. Sommer, R., & Paxson, V. (2010, May). Outside the closed world: On using machine learning for network intrusion detection. In 2010 IEEE Symposium on Security and Privacy (pp. 305–316). IEEE. https://doi.org/10.1109/SP.2010.25
5. Ghosh, R., & Raj, P. (2020). Artificial intelligence for software engineering: Challenges and opportunities. arXiv preprint arXiv:2006.03875. https://arxiv.org/abs/2006.03875
6. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., ... & Dennison, D. (2015). Hidden technical debt in machine learning systems. In Advances in Neural Information Processing Systems (NeurIPS), 28. https://papers.nips.cc/paper_files/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html
7. Varshney, K. R., & Alemzadeh, H. (2017). On the safety of machine learning: Cyber-physical systems, decision sciences, and data products. Big Data, 5(3), 246–255. https://doi.org/10.1089/big.2017.0007