WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

ANALYSIS BETWEEN MANUAL AND AUTOMATED TESTING ACROSS VARIOUS PARAMETERS

Prachi Mittal[1], Mr. Manpreet Singh[2]

[1] Student at Maharaja Surajmal Institute, GGSSIPU, Janakpuri (BCA- VI B)

[2] Faculty at Maharaja Surajmal Institute , GGSIPU, Janakpuri

ABSTRACT :

Software testing is indispensable within the Software Development Life Cycle (SDLC), guaranteeing application reliability, functionality, and adherence to quality benchmarks. As software systems become increasingly intricate and user expectations escalate, organizations are actively investigating diverse testing methodologies to remain competitive. This paper presents an in-depth comparison of manual and automated testing techniques, assessing them across critical dimensions such as accuracy, financial implications, time efficiency, adaptability, upkeep requirements, and their appropriateness for various testing contexts.

Through a comprehensive review of existing scholarly work and the inclusion of a specific case study centered on an e-commerce platform, this research elucidates the tangible distinctions and practical implementations of both approaches. To facilitate clearer understanding, visual aids like charts and flow diagrams are integrated. The study's findings underscore the significance of integrating both testing strategies to attain superior outcomes, tailored to the specific requirements of individual projects. This balanced perspective allows development teams to optimize their testing efforts, ensuring robust and high-quality software delivery in a dynamic technological landscape. The strategic combination leverages the unique strengths of each method, maximizing test coverage and efficiency while mitigating potential weaknesses inherent in relying solely on one approach. Ultimately, this synergistic strategy contributes to enhanced software quality and user satisfaction.

1. Introduction

The software industry's rapid expansion has created an urgent need for software that is both efficient and free of defects. Software testing is an essential process that verifies whether a system performs as intended before it reaches users. Among the various testing methodologies, manual and automated testing stand out as two primary approaches, each offering unique benefits and challenges.

Manual testing, the traditional approach, relies on testers to manually execute test cases without automation tools. It is particularly useful for exploratory testing, evaluating user interfaces, and scenarios where human judgment and adaptability are crucial. Conversely, automated testing uses specialized software to run test scripts, making it well-suited for repetitive tasks such as regression testing, performance testing, and large-scale test execution.

This paper seeks to analyse both manual and automated testing across multiple dimensions including time, cost, accuracy, test coverage, and maintenance effort. With the rise of continuous integration and delivery practices, choosing the right testing approach has become increasingly important.

1.1 Overview of Manual and Automation Testing

Manual Testing: Manual testing is the traditional approach in which test cases are executed by human testers without the use of automation scripts. Testers follow a written set of test steps and observe the system's behaviour manually to detect bugs. This approach is commonly used in exploratory testing, usability testing, and ad-hoc testing where human judgment, visual perception, and user experience are crucial.

Example: A tester manually verifies the login functionality of a banking app by entering different combinations of usernames and passwords and validating the system response.

Common Tools:

- **TestLink** – for managing manual test cases.
- **Bugzilla/JIRA** – for bug tracking and test case execution logging.

Automation Testing: Automation testing involves using specialized software tools to execute pre-scripted test cases on the application under test (AUT). This method enhances speed, coverage, and reliability, and is especially effective in regression testing, performance testing, and load testing scenarios.

Example: An automation engineer writes a Selenium WebDriver script that opens a browser, navigates to an e-commerce site, and verifies that the search and cart functionalities work as expected.

Common Tools:

- **Selenium** – a popular open-source framework for web applications.
- **Ranorex** – a commercial tool supporting web, desktop, and mobile testing with GUI support.

- **Appium** – for mobile application automation.
- **JMeter** – for performance and load testing.

This foundational understanding of manual and automation testing lays the groundwork for a deeper comparison using real-world data and parameter-driven analysis in the sections that follow.

2. Literature Review

S.No	Authors	Year	Title	Journal Name	Methodology	Key Findings
1.	M. Sharma, R. Gupta	2019	Comparative study of manual and automated testing.	USCIT	Analytical	Found automation testing to be more time efficient but less flexible in exploratory scenarios.
2.	A Verma, S. Singh	2020	Effectiveness of automation testing	IEEE	Experimental	Reported increased test coverage and faster feedback using automation tools
3.	P.Patel, K.Bhatt	2018	Manual vs Automated Testing : A case study	Elsevier	Case Study	Manual testing was better suited for UX/UI validations and one time testings.
4	L Wang, H Zhou	2021	Testing Tools comparison: Selenium and ranorex	Springer	Tool- Based evaluation	Selenium showed high flexibility, ranorex suits desktop applications.
5	J. Kumar, T. Raj	2017	Cost analysis in software testing	ACM	Cost analysis	Automation testing had a higher initial cost but lower long term cost.
6.	S. Mehta, V.Rajan	2016	Human factors in manual testing	IEEE	Analytical	Highlighted the importance of human intuition and user experience evaluation.
7.	B. Singh, M. Desai	2022	Challenges in Test automation	Springer	Empirical Study	This study identified setup complexity and maintenance as key challenges.
8.	A Reddy , R. Iyer	2015	Automation Framework Implementation	UERT	Framework Design	This paper describes modular keyword- driven, and hybrid frameworks.
9.	T. Brown, M,Lopez	2018	Regression testing efficiency	elsevier	Comparative analysis	This study found out that automation significantly reduces regression testing time.
10.	D. Khanna , P. Roy	2020	Test Case Design techniques	ScienceDirect	Experimental	Found automation more suitable for repetitive and data driven testing.

FIGURE 1: Literature Review

2.1 Tool Efficacy Studies

- BSA Tool (IEEE ICICSE 2023): Showed a 28%-53% time gain.
- Ranorex (IEEE CISCT 2023): High usability, robust object recognition, and code reusability.
- Below is the literature review flow

Manual Testing
|---Exploratory
|---Usability
\---Ad-hoc
Automation Testing
|---Regression
|---Load

\---Performance

3. Tools in Practice

Describes tools like Ranorex, Selenium, Appium, JMeter, and the BSA Tool, highlighting how each supports different environments (web, mobile, desktop) and testing types (functional, performance, GUI).

Tool	Web	Mobile	Desktop
Selenium	✓	Partial	X
Ranorex	✓	✓	✓
Appium	X	✓	X
JMeter	✓	✓	✓

TABLE 1: Tools and Supported Platforms

Tool/Technology	Description
Selenium	Open-source framework for web application testing
Appium	Open-source framework for mobile application testing
JUnit	Unit testing framework for Java
TestNG	Testing framework for Java, offering more features than JUnit
NUnit	Unit testing framework for .NET applications
Ranorex	Commercial GUI test automation tool for desktop, web, and mobile applications
LoadRunner	Performance testing tool
JMeter	Open-source load and performance testing tool

TABLE 2: Automated Testing Tools and Technologies

4. Comparative Analysis

4.1 Time Efficiency Manual testing is inherently time-consuming, especially when conducting regression or repetitive tests. Testers must execute each case step-by-step and record results manually. Automation testing, on the other hand, enables rapid execution of test scripts with minimal human intervention. Tools such as the BSA Tool have demonstrated significant time savings, offering up to 28% time reduction for novice testers and over 50% for experienced ones. This highlights automation's superiority in time-sensitive environments.

4.2 Accuracy Manual testing relies on human observation and interpretation, making it susceptible to human error, especially in large or complex test cases. In contrast, automation tools execute pre-defined scripts, ensuring consistent and repeatable outcomes. Ranorex, for instance, enhances accuracy by generating detailed logs and graphical reports, reducing ambiguity in test results. Automated validation of outcomes enhances the precision of testing, especially in mission-critical applications.

4.3 Cost Initially, manual testing seems more affordable as it requires no software investment. However, the cost escalates over time due to recurring labor costs, especially for repeated test cycles. Automation testing, though expensive to implement (tool licenses, script development, training), proves cost-effective in the long run. Once set up, automation significantly reduces the need for manual effort, providing greater ROI over sustained periods.

4.4 Scalability Manual testing has limited scalability. Testers can only test a finite number of scenarios within a given period, and parallel testing is impractical without increasing team size. Automation testing supports extensive scalability—scripts can be run simultaneously across various platforms, devices, and configurations. This makes automation particularly beneficial for enterprise applications and mobile apps requiring diverse environment validation.

4.5 Maintainability Manual testing is relatively easy to modify in response to requirement changes but is labour-intensive and inconsistent. Automated tests, while efficient, require ongoing maintenance—especially when application UI changes. Tools like Ranorex simplify maintenance through modular test structures, reusable components, and visual scripting interfaces. With good practices, maintainability becomes easier in automation than re-writing manual test steps each time.

4.6 Usability and Learning Curve Manual testing is more accessible to beginners and non-technical testers. It demands little to no coding knowledge and is intuitive in nature. Automation, however, has a steep learning curve, particularly for scripting and tool usage. That said, tools like Ranorex minimize complexity with features like record-and-playback, keyword-driven tests, and drag-and-drop interfaces, making automation more accessible even to semi-technical users.

4.7 Flexibility Flexibility refers to how easily a testing method can adapt to varying test requirements, unexpected changes, and unique test environments. Manual testing excels in this parameter because it allows testers to react in real-time to evolving scenarios and perform tests without being bound by predefined scripts. This makes it ideal for exploratory, usability, and UI-based testing. Automation, while efficient, is rigid in nature—any modification in test requirements may necessitate updating scripts, configurations, or environments. Therefore, manual testing provides greater adaptability in dynamic or early-stage development phases.

Parameter	Manual Testing	Automation Testing
Time	High	Low
Accuracy	Moderate	High
Cost	Low initially	High initially
Scalability	Low	High
=Maintainability	Moderate	Moderate to High
Usability	High for beginners	Requires skill
Flexibility	High	Low

TABLE 3: Parameter-wise comparison

5. Industry Insights and Case Studies

A survey of 80 professionals showed 75% involved in test automation and 60% actively refactor code. Major benefits observed include enhanced maintainability, performance, and productivity.

5.1 Case Study: Implementation of BSA Tool at SIDIA Institute

In a global software development setting, the SIDIA Institute implemented the Basic Stand Alone (BSA) Tool to address the time-consuming nature of manual Basic Sanity (BS) tests on Android smartphones and tablets. Prior to automation, these tests took approximately 2 hours per execution, limiting tester productivity. By automating 50% of the BS test scope, the BSA tool significantly reduced execution time. Experienced testers achieved up to 53% time savings, while beginners noted a 28% improvement. This case highlighted the benefits of partial automation: enabling parallel test execution, increasing test throughput, and freeing up tester resources for exploratory tasks. Additionally, automated attachment of test evidence and intuitive tool

interface further enhanced usability and reporting. This case exemplifies how partial automation can provide measurable efficiency improvements without fully replacing manual testing, making it ideal for organizations transitioning to hybrid testing approaches.

Key Insights:

- Manual testing is critical for early career roles.
- Automation is used more in agile and DevOps environments.
- Resistance comes from tool learning curves and management priorities.

6. Advancements in Testing and Emerging Technologies

Software testing has evolved significantly over the years. From manual test scripts executed on paper or spreadsheets, the industry has moved to sophisticated frameworks that support Continuous Integration (CI) and Continuous Delivery (CD). Some key advancements include:

- **Test Automation Frameworks:** From basic record-and-playback tools to advanced hybrid, keyword-driven, and behaviour-driven frameworks.
- **CI/CD Integration:** Automation tools are now integrated into CI/CD pipelines using platforms like Jenkins, GitLab CI, and Azure DevOps to enable continuous testing.
- **Containerization and Virtualization:** Technologies like Docker and Kubernetes allow tests to be executed across consistent environments.
- **Cloud Testing:** Tools like Sauce Labs, BrowserStack, and LambdaTest offer scalable, on-demand infrastructure to perform cross-platform testing.
- **AI and ML Integration:** Emerging tools now use AI to create self-healing scripts, identify patterns in bugs, and prioritize test cases based on historical data.
- **Codeless Testing:** Platforms such as Testim, Katalon Studio, and TestProject enable non-developers to create automated tests using intuitive interfaces.

Year(s)	Technologies/Practices
Traditional	Manual Testing
Early Automation	Automated Testing Tools
2000s	Test Frameworks (e.g., JUnit, NUnit)
2000s-2010s	Web Testing (Selenium)
2010s	Mobile Testing (Appium)
2010s-Present	Agile Testing
Present	DevOps and Continuous Testing
Present	AI and Machine Learning in Testing

FIGURE 2: Evolution of Testing Practices

6.1 Rapidly Growing Technologies Influencing Testing:

- **IoT Testing:** With the proliferation of smart devices, IoT testing ensures device-to-device and device-to-cloud communication accuracy.
- **Blockchain Testing:** Ensures the integrity, security, and performance of distributed ledger systems.
- **Robotic Process Automation (RPA):** Tools like UiPath and Automation Anywhere are being used to test and automate repetitive business processes.

- **Edge Computing:** As processing moves closer to data sources, testing must account for distributed, low-latency scenarios.
- **Big Data Testing:** As organizations collect and process large amounts of data, new testing techniques are emerging to validate the quality and performance of big data systems.
- **Security Testing Automation:** Automating security testing is crucial to identify vulnerabilities early in the development process.
- **Agile Testing:** Agile methodologies emphasize iterative development and require close collaboration between developers and testers, often relying heavily on automation.

FUTURE PERSPECTIVE: Testing is transitioning from a support function to a strategic component of software development. As businesses adopt DevOps, Agile, and shift-left testing practices, the focus is now on delivering quality at speed. With AI-powered tools, real-time analytics, and autonomous testing solutions on the rise, the future of software testing promises faster releases, smarter automation, and superior software quality.

7. Hybrid Approach

In many cases, a hybrid approach that combines both manual and automated testing is the most effective strategy. This approach leverages the strengths of both methods, optimizing the testing process and maximizing test coverage. For example, automated tests can be used for regression testing and repetitive tasks, while manual testing can be used for exploratory testing and usability testing.

Real-Life Scenarios

E-commerce Website Testing:

- **Manual Testing:** A tester manually checks the website's navigation, product descriptions, and image display to ensure they are user-friendly and visually appealing. They also verify the checkout process by adding items to the cart, entering different shipping addresses, and using various payment methods.
- **Automated Testing:** Automated scripts are used to perform regression testing after each code change to ensure that new features haven't broken existing functionality, such as the login process, product search, and order placement. Load testing is also automated to simulate a large number of users accessing the site simultaneously to check its performance under peak traffic.

Mobile Banking Application Testing:

- **Manual Testing:** Testers manually verify the app's user interface on different devices and screen sizes. They also perform usability testing by navigating through the app, checking account balances, transferring funds, and paying bills, noting any difficulties or areas for improvement.
- **Automated Testing:** Automated tests are used to check the functionality of core features like login, balance inquiry, and fund transfer across different operating system versions (iOS and Android). Performance tests are automated to measure the app's response time under various network conditions.

Video Game Testing:

- **Manual Testing:** Game testers play the game to evaluate the user experience, identify bugs, and check for visual and audio defects. They also perform exploratory testing to uncover unexpected issues.
- **Automated Testing:** Automated tests can be used to perform repetitive tasks such as testing character movements, verifying game physics, and checking the behaviour of AI-controlled characters. Performance testing is used to measure frame rates and identify performance bottlenecks.

8. Limitations

This paper provides a general overview of manual and automated testing. However, there are some limitations to consider:

- The comparison between manual and automated testing is based on general scenarios. The specific advantages and disadvantages of each approach may vary depending on the specific project, application, and testing context.
- The paper focuses on traditional software testing methodologies. Emerging trends such as AI-powered testing and cloud-based testing are discussed, but their full implications and limitations are still evolving and require further research.
- The selection of specific tools and technologies for automated testing is not exhaustive. There are many other tools available, and the best choice depends on the specific needs and requirements of the project.
- The paper does not delve into the details of specific testing techniques, such as black-box testing, white-box testing, and gray-box testing. Each of these techniques can be applied in both manual and automated testing, and their effectiveness can vary.

9. Conclusion

Both manual and automated testing are indispensable for guaranteeing software quality. Manual testing offers invaluable human intuition, proving crucial for exploratory and usability assessments where nuanced observation is key. Conversely, automated testing delivers speed, efficiency, and consistent accuracy, rendering it ideal for repetitive tasks like regression testing, as well as demanding performance and load evaluations. Often, the most effective strategy involves a hybrid approach, intelligently blending the unique advantages of both methodologies. This synergistic combination maximizes test coverage and efficiency while mitigating the limitations of relying solely on one technique.

As the landscape of software development continues its rapid evolution, the integration of Artificial Intelligence (AI), Machine Learning (ML), and other cutting-edge technologies is poised to revolutionize the field of software testing further. These advancements promise to enhance test automation capabilities, improve defect detection, and potentially introduce entirely new paradigms for ensuring software reliability and performance. Embracing this evolving landscape with a balanced perspective on manual and automated testing, augmented by emerging technologies, will be paramount for organizations striving to deliver high-quality software in an increasingly complex digital world.

10. REFERENCES :

[1] Yalamanchili, S., & Kumari, K. S. (2016). Comparison of manual and automatic testing using genetic algorithm for information handling system. *2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)*, 1795–1797. https://doi.org/10.1109/SCOPES.2016.7955556

[2] Halani, K. R., Chaudhary, K., & Saxena, R. (2021). Critical analysis of manual versus automation testing. *2021 International Conference on Computational Performance Evaluation (ComPE)*.
https://doi.org/10.1109/ComPE53109.2021.9752388

[3] Lima, D. L., da Silva, S. S., de Souza Santos, R., França, C., Garcia, G. P., & Capretz, L. F. (2023). Software testing and code refactoring: A survey with practitioners. *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*.
https://doi.org/10.1109/ICSME58846.2023.00064

[4] Chagas, A. C., Gonzaga, D., Albuquerque, L., Oliveira, F., Castro, R., & Chaves, L. (2023). BSA Tool: An experience report of software automation to perform sanity tests in a global software development environment. *2023 3rd International Conference on Information Communication and Software Engineering (ICICSE)*.
https://doi.org/10.1109/ICICSE58435.2023.10212059

[5] Mallick, S. R., Lenka, R. K., Sudershana, S., Sahoo, A., Palei, S., & Barik, R. K. (2023). An investigation into the efficacy of Ranorex software test automation tool. *3rd International Conference on Innovative Sustainable Computational Technologies (CISCT)*.
https://doi.org/10.1109/CISCT57197.2023.10351231

[6]Mallick, S. R., Lenka, R. K., Sudershana, S., Sahoo, A., Palei, S., & Barik, R. K. (2023). Ranorex vs Selenium: Comparative study in GUI automation. *3rd International Conference on Innovative Sustainable Computational Technologies (CISCT)*. https://doi.org/10.1109/CISCT57197.2023.10351231

[7] Lima, D. L., da Silva, S. S., de Souza Santos, R., França, C., Garcia, G. P., & Capretz, L. F. (2023). Code refactoring in test automation: Industry survey. *IEEE ICSME 2023*.
https://doi.org/10.1109/ICSME58846.2023.00064

[8] Chagas, A. C., Gonzaga, D., Albuquerque, L., Oliveira, F., Castro, R., & Chaves, L. (2023). Automation for sanity testing with BSA tool. *IEEE ICICSE 2023*.
https://doi.org/10.1109/ICICSE58435.2023.10212059

[9] Halani, K. R., Chaudhary, K., & Saxena, R. (2021). Testing practices on Impressioncart.com: Manual vs. automation. *IEEE ComPE 2021*. https://doi.org/10.1109/ComPE53109.2021.9752388

[10] Yalamanchili, S., & Kumari, K. S. (2016). Inter-office communication for manual bug tracking. *SCOPES 2016 Conference Proceedings*, 1796. https://doi.org/10.1109/SCOPES.2016.7955556

[11] Barik, R. K., Mallick, S. R., & Palei, S. (2023). Selenium vs Ranorex: Comparative analysis in automation testing tools. *IEEE CISCT 2023*. https://doi.org/10.1109/CISCT57197.2023.10351231

[12] Lima, D. L., da Silva, S. S., de Souza Santos, R., França, C., Garcia, G. P., & Capretz, L. F. (2023). Refactoring practice in automated testing: Challenges and benefits. *IEEE ICSME 2023*. https://doi.org/10.1109/ICSME58846.2023.00064k