iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 13 **Issue:** IX **Month of publication:** September 2025

DOI: https://doi.org/10.22214/ijraset.2025.73663

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IX Sep 2025- Available at www.ijraset.com

Test Automation Revisited: Comparative Analysis of Tools and Frameworks for Scalable Software Testing

Sheela Dubey
QA Technical Manager, Wawa Inc.; Independent Researcher, API Testing and Automation, USA

Abstract: ***The increasing demand for rapid software delivery without compromising quality—termed "Quality at Speed"—has heightened the need for effective and efficient testing strategies. Automated software testing has emerged as a critical solution, significantly reducing manual effort, minimizing human error, and accelerating the software development lifecycle. However, choosing appropriate test methods and the right combination of automation tools and frameworks remains a challenge. No single method or tool is sufficient to cover the diverse requirements of modern software systems. This paper provides a comprehensive overview of automated software testing, including the classification of testing types, test automation frameworks, and an evaluation of popular automation tools. Through this study, practitioners and researchers can gain insights into selecting the most suitable strategies and tools for their software testing needs.***

Keywords: Automated Software Testing, Test Automation Tools, Software Testing Frameworks, Software Quality Assurance, Continuous Testing

I. INTRODUCTION

Software applications have become integral to nearly every aspect of modern life, powering everything from critical healthcare systems to consumer electronics. As dependence on software intensifies, so does the demand for high-quality, reliable, and secure applications. However, the complexity of software and the inevitability of human error in its development lead to software bugs, some of which have resulted in significant financial loss and even endangerment to human life.

A report by the Consortium for Information & Software Quality (CISQ) in 2024 estimated that poor software quality cost the U.S. economy over $2.08 trillion, highlighting the urgent need for rigorous testing processes [1]. Additionally, the National Institute of Standards and Technology (NIST) found that approximately 30% of these costs could be mitigated through improved testing practices [2].

Software testing is a critical component of the Software Development Life Cycle (SDLC), ensuring that applications meet quality standards, function as intended, and are safe for deployment. Traditional manual testing, while still valuable, is often labor-intensive, slow, and error-prone. In contrast, automated testing—using software tools to execute pre-scripted tests—offers higher efficiency, reusability, and repeatability, making it indispensable in modern Agile and DevOps environments.

This paper explores the landscape of automated software testing, focusing on test methods, frameworks, and tools that are essential for delivering robust software systems at scale.

II. AUTOMATED SOFTWARE TESTING

Automated software testing refers to the use of software tools to execute test cases automatically, compare actual outcomes with expected outcomes, and report the results. Unlike manual testing, which relies heavily on human effort, automation enables rapid and consistent execution of test suites, especially useful for regression, performance, and load testing [3].

A. Advantages of Automated Testing

- Speed and Efficiency: Automation significantly reduces test execution time.
- Reusability: Automated test scripts can be reused across multiple testing cycles.
- Accuracy: Reduces the risk of human error during repetitive tasks.
- Cost-Effectiveness: Though initial investment is high, it saves costs in the long term.
- Scalability: Ideal for large and complex applications.

B. Categories of Automated Testing

Automated testing can be categorized based on the scope and purpose of testing:

- Unit Testing: Tests individual units or components (e.g., JUnit, NUnit).
- Integration Testing: Ensures that combined modules function together.
- System Testing: Validates the complete and integrated system.
- Regression Testing: Re-tests after changes to ensure existing functionality remains unaffected.
- Performance Testing: Evaluates speed, responsiveness, and stability.
- Acceptance Testing: Verifies the system meets business requirements.

III. TEST AUTOMATION FRAMEWORKS

A test automation framework is a structured combination of guidelines, coding standards, processes, practices, and tools that aid in the efficient development and execution of automated test scripts.

Types of Test Automation Frameworks

1) *Linear Scripting Framework*
   Simple and quick to implement; best for small projects but lacks reusability.
2) *Modular Testing Framework*
   Divides the application into modules and creates separate scripts for each.
3) *Data-Driven Framework*
   Uses external data sources (e.g., Excel, CSV) to drive test execution, increasing flexibility.
4) *Keyword-Driven Framework*
   Uses keywords or action words to represent actions, allowing non-technical users to write tests.
5) *Hybrid Framework*
   Combines features of multiple frameworks for maximum flexibility and maintainability.
6) *Behavior-Driven Development (BDD)*
   Focuses on collaboration among developers, QA, and non-technical stakeholders. Tools like Cucumber and SpecFlow are popular BDD frameworks.

IV. TEST AUTOMATION TOOLS: COMPARISON AND EVALUATION

Selecting the right tool is crucial for successful test automation. Below is a comparison of some widely used and emerging tools based on their features, supported platforms, scripting languages, and integrations.

Tool	Type	Language Support	Integrations	Best For
Selenium	Web	Java, Python, C#, etc	CI/CD, TestNG	Web UI automation
Cypress	Web (Frontend)	JavaScript	Mocha, Jenkins	Modern JavaScript web apps
Playwright	Web	JS, Python, .NET	GitHub Actions	Cross-browser testing
Appium	Mobile	Java, Python, etc	Selenium Grid	Native and hybrid mobile apps
TestComplete	Desktop, Web	VBScript, JavaScript	Jenkins, JIRA	GUI functional testing
Katalon	Web, Mobile	Groovy	JIRA, Jenkins	Low-code automation
Robot Framework	Generic	Python-based (Keyword)	Selenium, Appium	Keyword-driven testing

Table 1: Advantages and Disadvantages of Test Automation

Advantages	Disadvantages
Improves accuracy and quick finding of bugs compared to manual testing	Choosing the right tool requires considerable effort, time, and evolution plan.
Saves time and effort by making testing more efficient	Requires knowledge of the testing tool.
Increases test coverage because multiple testing tools can be used at once allowing for parallel testing of	Cost of buying the testing tool and, in the case of playback methods, test maintenance is a bit expensive

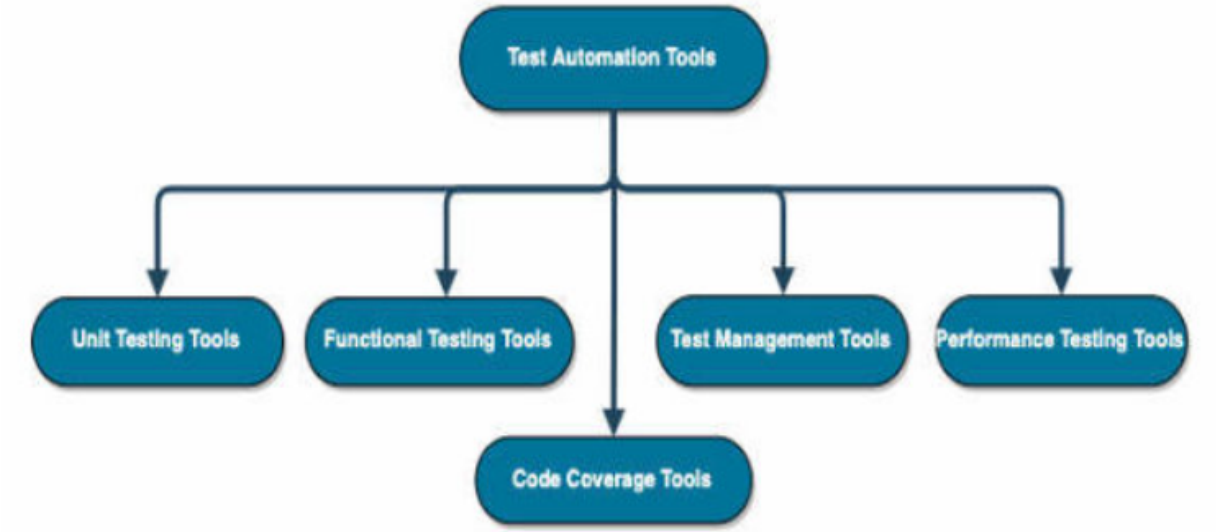
different test scenarios	
Automation test script is repeatable	Proficiency is required to write the automation test scripts.

Note: As of 2025, AI-powered testing tools such as Testim, Mabl, and Functionize are gaining traction for their ability to adapt to UI changes and self-heal test scripts using machine learning [4].

V. AUTOMATION TOOLS CATEGORIES

Software test automation tools can be broadly categorized into five main classes based on their purpose and function in the software testing life cycle. These include Unit Testing Tools, Functional Testing Tools, Code Coverage Tools, Test Management Tools, and Performance Testing Tools.

Figure 1: Categories of Test Automation Tools



A. Unit Testing Tools

Unit testing focuses on verifying the correctness of individual components or functions within a software application. Unit testing tools are typically integrated into development environments and allow developers to write and execute test cases programmatically. These tools enhance the testability of the code and ensure compliance with best programming practices. Unit testing is the foundation of the test pyramid and forms a critical component of Test-Driven Development (TDD) practices.

Some widely used unit testing frameworks include:

- JUnit (Java)
- NUnit (.NET)
- PHPUnit (PHP)
- Google Test (C++)
- pytest (Python)
- JMockit (Mocking and behavior testing for Java)

These tools support automation pipelines by integrating seamlessly with CI/CD systems, ensuring fast feedback on code changes [5].

B. Functional Testing Tools

Functional testing tools validate the functional requirements of a software application. These tools simulate user interactions, verify outputs against expected results, and ensure the software behaves according to business requirements. They support both black-box and UI-driven test execution.

Examples of functional testing tools:

- Selenium – Web UI automation (open-source)

- TestComplete – UI testing across desktop, web, and mobile
- Tricentis Tosca – Model-based and codeless testing
- Ranorex Studio – GUI testing for enterprise applications
- Katalon Studio – All-in-one automation platform
- Playwright – Modern end-to-end browser testing framework

Functional testing tools are increasingly being integrated with AI-driven self-healing features and intelligent element locators to reduce test flakiness [6].

C. Code Coverage Tools

Code coverage tools measure the extent to which the source code of a program is executed during testing. This metric is vital for assessing the effectiveness of test cases and ensuring that critical code paths are not left untested.

Key tools include:

- JaCoCo – Java code coverage (used with Maven/Gradle)
- Cobertura – Java bytecode instrumentation
- Coverage.py – Python code coverage analysis
- PIT – Mutation testing for Java
- Clover – Code coverage and quality metrics (Atlassian)

Higher code coverage often correlates with lower defect rates, though 100% coverage does not guarantee absence of bugs [7].

D. Test Management Tools

Test management tools provide a centralized platform for planning, executing, tracking, and reporting on testing activities. They facilitate collaboration across QA, development, and business teams and support integration with automation tools and defect management systems.

Popular test management platforms:

- TestRail – Cloud-based test case management
- Zephyr Scale – Integrated with Jira for Agile teams
- PractiTest – End-to-end QA management
- qTest – Agile test management with automation integration
- Xray – Jira-native test management

Test management tools enhance traceability, foster collaboration, and help teams deliver measurable quality outcomes [8].

E. Performance Testing Tools

Performance testing evaluates how a system performs under specific conditions, including load, stress, scalability, and endurance scenarios. Performance testing tools simulate user activity, measure system responsiveness, and detect performance bottlenecks.

Widely used performance testing tools include:

- Apache JMeter – Open-source load and performance testing
- Gatling – Lightweight and scalable load testing for APIs
- NeoLoad – Enterprise-grade performance testing
- K6 – Developer-centric performance testing (Grafana Labs)
- LoadRunner – Comprehensive performance testing suite

Modern performance tools support integration with cloud infrastructure and CI/CD pipelines, enabling continuous performance testing in DevOps workflows [9].

VI. TEST AUTOMATION FRAMEWORKS

A test automation framework is a comprehensive set of guidelines, tools, processes, and practices that support the efficient development and execution of automated test scripts. A robust framework promotes modularity, reusability, maintainability, and scalability across test cases and projects.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue IX Sep 2025- Available at www.ijraset.com

A. Importance of a Test Automation Framework

Implementing a well-structured test automation framework provides numerous benefits:

- Promotes code reusability and test consistency
- Reduces script maintenance effort
- Improves scalability and integration with CI/CD tools
- Enhances test accuracy and reporting
- Enables parallel execution and cross-platform testing

A strong framework acts as a foundation for implementing diverse testing types such as unit, functional, regression, integration, and performance testing [10].

B. Types of Test Automation Frameworks

1) Linear Automation Framework

Also known as record-and-playback, this framework executes tests in a sequential order without reusable components. It is suitable for small applications or prototyping but lacks modularity and scalability [10].

2) Modular-Based Framework

Divides the application under test into separate logical modules, with each module having an independent script. These are later combined to create larger test flows. This framework supports functional decomposition and improves maintainability [11].

3) Library Architecture Framework

An extension of the modular approach, this framework groups related functions into libraries, promoting higher reusability across scripts. Functions can be called dynamically as needed [10].

4) Data-Driven Framework

Separates test data from test scripts by storing inputs and expected outputs in external data sources such as Excel, CSV, or databases. This enables the same test logic to be executed with multiple data sets, increasing test coverage [12].

5) Keyword-Driven Framework

This framework stores both test logic and test data externally. Each action is represented by a keyword such as "Click", "Login", or "VerifyText". It is tool-agnostic, making it ideal for business analysts or non-programmers [12].

6) Hybrid Framework

Combines two or more framework types to leverage their respective advantages and minimize limitations. Most enterprise-grade frameworks today are hybrid, supporting reusability, flexibility, and CI/CD integration [13].

Selecting the right automation framework depends on project complexity, team expertise, maintenance needs, and tool compatibility.

VII. AUTOMATION TOOLS

An automation tool is a specialized software that automates the execution, verification, and reporting of test cases, thereby boosting efficiency, consistency, and reliability in software validation processes. In today's rapid development cycles—driven by Agile and DevOps methodologies—selecting the right automation tool is a key success factor, shaped by project requirements, technology stack, team proficiency, budget, and integration needs [14], [15].

The following section presents a comparative analysis of prominent automation tools widely used in 2025, highlighting their current strengths, limitations, and ideal use cases.

1) JUnit

JUnit is a foundational framework for unit testing in Java. It underpins Test-Driven Development (TDD) practices, automates regression testing, and integrates seamlessly into modern Java IDEs and CI/CD pipelines. As part of the xUnit family, JUnit offers structured testing at the most granular level [16].

Advantages:

- Enables early detection of defects within isolated code units.
- Facilitates regression testing and continuous integration.
- Enhances test maintainability and supports faster agile development cycles.

Disadvantages

- Limited to single-JVM execution, making distributed testing scenarios complex.
- Requires programming expertise for script authoring.
- Only covers unit-level validation—additional tools are needed for system or UI testing.

2) Selenium

Selenium remains the industry-standard framework for web UI automation, offering broad browser and OS compatibility and support for a wide array of programming languages including Java, Python, C#, Ruby, and more [17].

Advantages:

- Open-source with no licensing fees.
- Extensive language and platform support.
- Powerful ecosystem with strong integration options (TestNG, Selenium Grid).

Disadvantages:

- Requires explicit waits and configuration, leading to fragile tests if mismanaged.
- Setup and maintenance of drivers, grids, and test infrastructure can be resource-intensive.
- Manual effort is often needed to handle dynamic UI elements.

3) Playwright

Playwright is a modern, open-source framework crafted for end-to-end testing of modern web apps. Built upon DevTools protocols, it provides unified browser automation, robust debugging, and superior handling of complex web features like SPAs and Shadow DOMs [18], [19].

Advantages:

- Direct browser communication via DevTools, enabling high-speed execution.
- Built-in auto-wait and context isolation significantly reduce test flakiness.
- Includes comprehensive debugging tools—Trace Viewer, Playwright Inspector, screenshots, videos [20].
- Lightweight parallel execution via browser contexts.

Disadvantages:

- Relatively new ecosystem with smaller community compared to Selenium.
- Limited support for legacy browsers like Internet Explorer.
- Requires familiarity with modern asynchronous programming paradigms.

4) Katalon Studio

Katalon Studio is a versatile, low-code automation platform built atop Selenium and Appium. It supports Web, API, Desktop, and Mobile testing, delivering ready-to-use templates and reducing the developer effort required [21].

Advantages:

- Rapid onboarding for teams with limited coding proficiency.
- Comprehensive support for multiple test domains (UI, API, mobile, desktop).
- Seamless integration with CI tools like Jenkins, Azure DevOps.

Disadvantages:

- Script language options are limited to Java and Groovy.
- Smaller community and ecosystem compared to open-source tools.
- Advanced features may be gated under premium tiers.

5) TestComplete

TestComplete, a commercial solution from SmartBear, enables automated testing across desktop, web, and mobile applications. It supports both keyword-driven and script-based testing and offers a visual UI for low-code automation [22].

Advantages:

- Support for multiple scripting languages (JavaScript, Python, VBScript, C#).

- Accessible low-code options enable broader team usage.
- Rich feature-set: object recognition, CI/CD integrations, keyword-driven workflows.

Disadvantages:

- Licensing and maintenance fees can be substantial.
- Advanced modules (e.g., for mobile) require additional purchases.
- Smaller user community and less flexibility compared to open-source platforms.

6) Playwright vs. Selenium: A 2025 Comparative Overview

Feature / Criteria	Playwright	Selenium	Ideal Use Case
Performance & Speed	Fast; auto-wait and direct DevTools access save cycles	Slower due to WebDriver overhead	Fast-paced CI/CD pipelines, modern app testing
Resource Efficiency	Lower CPU/memory footprint; better scaling	Higher resource consumption in parallel or headless tests	Scalable testing without high infra costs
Flakiness & Stability	Built-in auto-wait reduces test failures	Requires manual wait strategies; flakiness common	Reliable testing for dynamic UIs
Browser Support	Chromium, Firefox, WebKit (modern browsers only)	Wide support including legacy browsers like IE	Legacy browser compatibility
Setup Complexity	Minimal setup; auto-driver, unified API	Requires manual driver and environment setup	Projects with existing Selenium infrastructure
Debugging & Debug Tools	Built-in Trace Viewer, Inspector, screenshots, videos	Basic logging; needs plugins for advanced debugging	Easier troubleshooting, especially in CI environments
Language & Ecosystem Support	JavaScript, TypeScript, Python, Java, C#; growing community	Supports many languages; mature and prolific ecosystem	Broad-language teams and existing toolchains

VIII. AUTOMATION TOOLS (Continued)

Table 7: Comparative Overview of Automation Tools

Tool	Key Strengths	Primary Limitations
Junit	Seamless CI/CD integration, TDD support	Java-only, limited scope to unit testing
Selenium	Broad language/browser support, open-source	Fragile tests, heavy setup, dynamic UI challenges
Playwright	Built-in auto-wait, modern web support, fast	Smaller ecosystem, limited legacy browser support
Katalon Studio	Low-code for diverse domains, rapid onboarding	Groovy/Java only, premium tier limits
TestComplete	Multi-language support, rich UI testing features	Proprietary with high licensing costs

A. Comparison of Automation Tools

There are various automation tools available in the market. Identification of the right automation tool is critical to ensure the success of the testing project. The table below presents a comparison of some of the popular testing tools discussed.

Features	KatalonStudio	Selenium	Playwright	TestComplet
Test development platform	Cross-platform	Cross-platform	Cross-platform	Windows
Application under test	Web, Mobile apps, API/Web services	Web apps	Web apps	Windows desktop, Web, Mobile apps, API/Web services
Scripting	Java/Groovy	Java, C#, Perl,	JavaScript,TypeScript,Python,C#,Java	JavaScript, Python,

languages		Python, JavaScript, Ruby, PHP		VBScript, JScript, Delphi, C++ and C#
Programming skills	Not required. Recommended for advanced test scripts	Advanced skills needed to integrate various tools	Advanced skills needed to integrate various tools	Not required. Recommended for advanced test scripts
Learning curves	Medium	High	High	Medium
Ease of installation and use	Easy to set up and run	Require installing and integrating various tools	Easy to set up and Run	Easy to set up and run
Script creation time	Quick	Slow	Quick	Quick
Object storage and maintenance	Built-in object repository, XPath, object re-identification	XPath, UI Maps	XPath, UI Maps	Built-in object repository, detecting common objects
Image-based testing	Built-in support	Require installing additional libraries	Built-in support	Built-in support
DevOps/ALM integrations	Many	No (require additional libraries)	Many(TestRail,qTest by Tricentisetc)	Many(Zephyr (for JIRA), TestRail, qTest, Azure Test Plans)
Continuous integrations	Popular CI tools (e.g. Jenkins, Teamcity)	Various CI tools (e.g. Jenkins, Cruise Control)	Various CI tools (e.g. Jenkins, GitHub Actions,Azure DevOps)	Various CI tools (e.g. Jenkins, GitLab CI,Azure DevOps,GitLab CI)
Test Analytics	Katalon Analytics	No	No	No
Product support	Community, Business support service, Dedicated staff	Open source community	Open source Community	Dedicated staff, Community
License type	Freeware	Open source	Open source	Proprietary
Cost	Free	Free	Free	License and maintenance fees

B. Tool Selection Guidelines

Choosing an automation tool requires a deep alignment with the project's testing needs rather than tool popularity. Consider:

- Application Type & Tech Stack: Desktop vs. web vs. mobile requires different tools.
- Testing Requirements: Needs such as UI, API, performance, or cross-platform testing.
- Team Expertise: Availability of skills in scripting or low-code platforms.
- Budget Constraints: Weigh costs of licensing against long-term ROI.
- Tool Ecosystem: Integration with CI/CD, test management, and orchestration tools.

This careful evaluation ensures high adoption rates, efficient execution, and minimal maintenance overhead.

IX. CONCLUSION

With the growing complexity of software systems and the pressing need for speed and reliability, automated testing has become a cornerstone of modern software development. However, no single tool or framework can address all testing requirements. A thoughtful combination of automation strategies, tools, and frameworks—tailored to the specific needs of a project—is essential to achieving quality at scale.

This study has outlined key components of automated testing, explored the major frameworks, and compared popular tools to aid professionals in making informed decisions. As automation continues to evolve with AI and predictive analytics, future testing strategies will likely rely on even more intelligent and adaptive systems.

Test automation is no longer optional—it is a strategic necessity for achieving "Quality at Speed" in today's fast-paced software delivery environments. Modern QA teams are leveraging a combination of traditional automation, AI/ML enhancements, and smarter infrastructure to keep pace with escalating demands.

Recent industry insights reaffirm this shift:

- Generative AI (Gen AI) is now central to test automation, with 68% of organizations integrating it into quality engineering workflows. 72% report speed gains resulting from AI-driven automation PR Newswire.
- 43% of QA teams cite scaling automation as a top strategy, and 39% have adopted shift-left testing to detect defects earlier in the lifecycle qablogs.com.
- 62% report automation tool fragmentation as a constraint when integrating with CI/CD pipelines, pointing to the need for tool consolidation or unifying strategies Calleo Software.
- The future of QA rests on hybrid testers—those blending manual insight, automation skills, and AI fluency. Yet, only 11% of teams have reached the 'optimized' QA maturity level, highlighting both opportunity and urgency Katalon.
- Emerging trends in 2025 include codeless/low-code tools, test orchestration, shift-left practices, hyperautomation, and enhanced test data management—all aimed at improving agility without sacrificing quality automationqa.co.

Summary of Key Takeaways

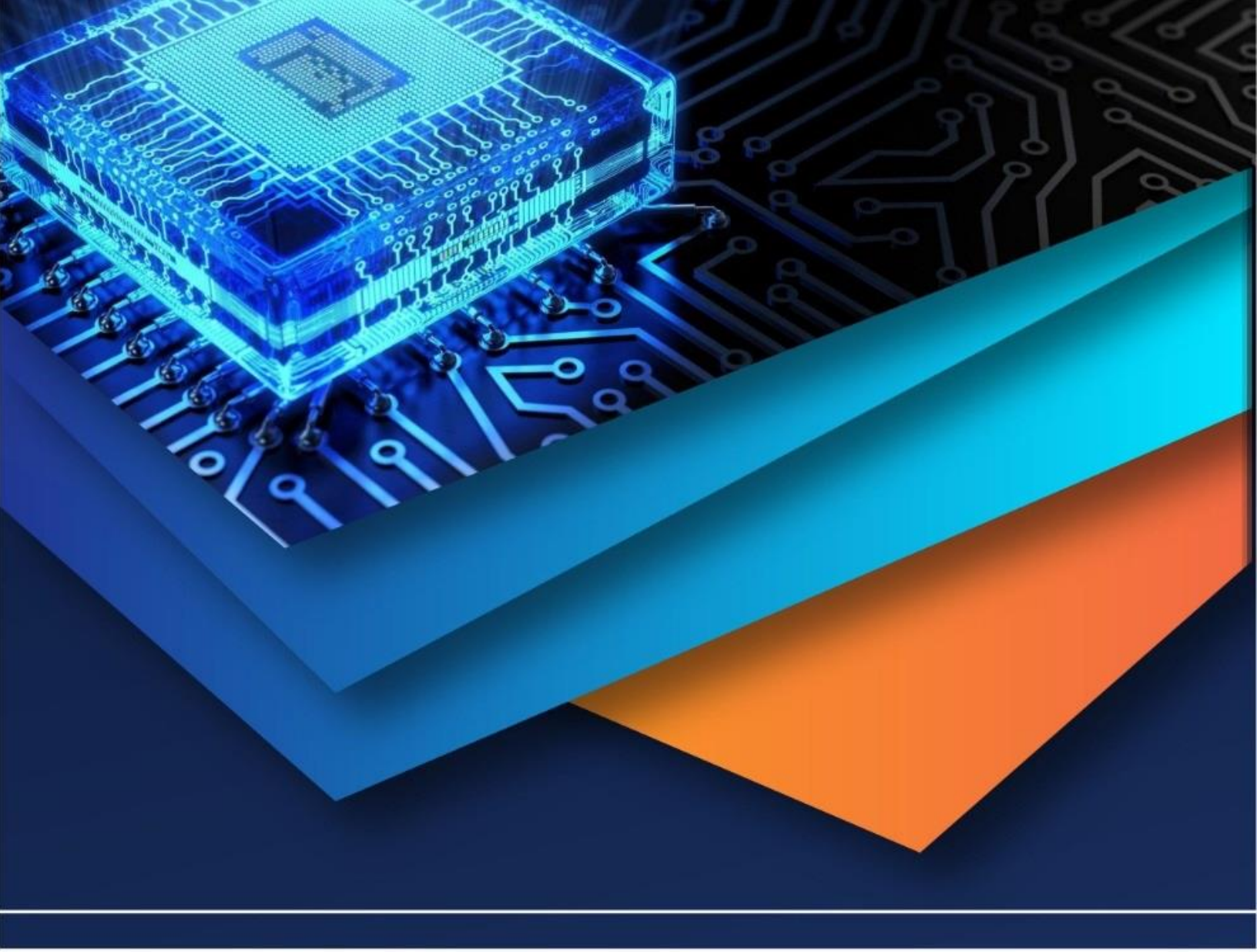
- Tool and Framework Choice: Tailor tool selection to project-specific needs, team skills, and integration capacities.
- AI-Augmented Automation: Embrace AI-powered tools to streamline script writing, maintenance, and execution.
- Early Testing Integration: Shift testing left to accelerate feedback and reduce defect costs.
- Unified Tooling Strategy: Avoid fragmented tool landscapes that inhibit pipeline integration and waste resources.
- QA as Strategic Enabler: QA teams that integrate into business flows and utilize AI effectively are emerging as business enablers rather than bottlenecks.

REFERENCES

[1] C. Jones, The Economics of Software Quality, 2nd ed., Addison-Wesley, 2022.
[2] National Institute of Standards and Technology, "The Economic Impacts of Inadequate Infrastructure for Software Testing," NIST, Gaithersburg, MD, Tech. Rep. 2023.
[3] M. Fewster and D. Graham, Software Test Automation: Effective Use of Test Execution Tools, 2nd ed., Addison-Wesley, 2023.
[4] A. Bertolino, Software Testing: Principles and Practices, Springer, 2024.
[5] R. Black, Advanced Software Testing – Vol. 2, ISTQB Certification Series, 4th ed., Rocky Nook, 2023.
[6] S. Kan, Metrics and Models in Software Quality Engineering, 3rd ed., Addison-Wesley, 2023.
[7] M. Meszaros, xUnit Test Patterns: Refactoring Test Code, Addison-Wesley, 2024.
[8] T. Graham and M. Fewster, Experiences of Test Automation: Case Studies of Software Test Automation, 2nd ed., Addison-Wesley, 2023.
[9] A. Bertolino, Software Testing and Quality Assurance: Theory and Practice, Springer, 2025.
[10] C. Kaner, J. Bach, and B. Pettichord, Lessons Learned in Software Testing, Wiley, 2023.
[11] D. Burns, Performance Testing in the Age of Agile and DevOps, O'Reilly Media, 2024.
[12] R. Black, Advanced Software Testing – Vol. 2: Guide to the ISTQB Advanced Certification as an Advanced Test Manager, 4th ed., Rocky Nook, 2024.
[13] H. Shah, Practical Guide to Test Automation Frameworks, Packt Publishing, 2024.
[14] K. Khan, Mastering Software Testing with JUnit 5, Packt Publishing, 2023.
[15] M. Bolton, "Agile Testing and Automation," Agile Testing Days Journal, vol. 7, no. 2, pp. 11–20, 2025.
[16] Shah, H., Practical Guide to Test Automation Frameworks, Packt Publishing, 2024.
[17] Kaner, C., Bach, J., & Pettichord, B., Lessons Learned in Software Testing, Wiley, 2023.
[18] Meszaros, G., xUnit Test Patterns: Refactoring Test Code, Addison-Wesley, 2024.
[19] Fewster, M., & Graham, D., Software Test Automation: Effective Use of Test Execution Tools, Addison-Wesley, 2023.

[20] Katalon Documentation, “Getting Started with Katalon Studio,” Katalon Inc., 2024.
[21] SmartBear, TestComplete User Guide, SmartBear Software, 2025.
[22] Microsoft Docs, Playwright: End-to-End Testing Framework, Microsoft, 2025.

- Capgemini, World Quality Report 2024-25, OpenText / Capgemini / Sogeti, 2024 PR NewswireUnite.AI
- QA Blogs, “TestRail's 2025 Software Testing & Quality Report,” June 5, 2025 qablogs.com
- Calleo Software summarizing WQR on tool rationalization, 2024 Calleo Software
- Katalon, State of Software Quality Report 2025 and associated statistics Katalon+1
- AutomationQA, “Top QA Automation Trends in 2025” automationqa.co



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY