Empirical Analysis of Widely Used Website Automated Testing Tools

Balqees Sani[1,*], Dr. Sadaqat Jan [1]

[1]Department of computer software engineering, University of Engineering and Technology, Mardan, 23200, Pakistan

Abstract

In today's software development, achieving product quality while minimising cost and time is critical. Automated testing is crucial to attaining these goals by lowering inspection efforts and discovering faults more effectively. This paper compares widely used automated testing tools, such as Selenium, Appium, JUnit (Java Unit), Test Next Generation (TestNG), Jenkins, Cucumber, LoadRunner, Katalon Studio, Simple Object Access Protocol User Interface (SoapUI), and TestComplete, based on functionality, ease of use, platform compatibility, and integration capabilities. Our findings show that no single tool is inherently superior, with each excelling in certain areas such as online, mobile, Application Programming Interface (API), or performance testing. While Selenium and Appium are the dominant online and mobile testing frameworks, TestComplete and Katalon Studio offer complete, user-friendly cross-platform testing solutions. Despite the benefits of automation, obstacles such as tool maintenance, scalability, and cost issues remain. The report finishes with advice for picking the best tool for the project and offers potential approaches for enhancing testing frameworks, such as AI-driven optimisation, cloud-based testing, and greater Continuous Integration/ Continuous Deployment (CI/CD) integration. This study offers useful information for developers and testers looking to optimise their testing methods and increase software quality.

Received on 20 April 2024; accepted on 01 September 2024; published on 10 October 2024

Keywords: Automated testing tools, Selenium, performance testing, Cloud-based testing

doi:10.4108/airo.7285

1. Introduction

Software testing plays a critical role in ensuring the quality, reliability, and functionality of software products [1]. It is an essential step in the software development lifecycle that helps identify errors, defects, and vulnerabilities before the software is released. Testing not only verifies that the software behaves as expected but also reduces risks and improves user satis-faction. Although it is impossible to achieve 100% testing coverage, it is vital to conduct thorough testing to ensure that the software can handle real-world environments, which differ significantly from development environments [2].

Traditionally, software testing was conducted manually by human testers who would execute test cases, evaluate performance, and identify defects [[3], [4]]. While manual testing has its benefits, such as mimicking real-world user interactions, it can be time-consuming and prone to human error, especially for large or complex projects [5]. In recent years, automated testing has gained popularity as a more efficient and scalable alternative. Automated testing tools allow for faster test execution, increased repeatability, and the ability to test across multiple quality attributes, such as performance, security, and reliability [6].

The adoption of automated testing tools has revolutionized the testing process, providing significant benefits in terms of speed, accuracy, and resource efficiency. Tools like Selenium and TestComplete enable developers and testers to automate repetitive tasks, such as regression and unit testing, allowing for more comprehensive and frequent testing [7]. These tools can evaluate various quality factors of a website, including response time, performance, and security, making them essential for modern software development practices.

*Corresponding author. Email: mrsohail236@gmail.com

However, no single tool is capable of testing all quality attributes, as each has its own strengths and limitations.

In recent years, object detection has gained prominence in fields such as computer vision, artificial intelligence, and automation, making it essential to ensure the reliability and accuracy of object detection systems [8]. Although automated testing tools are primarily designed to evaluate software functionality, performance, and user interfaces, they can be adapted to test object detection models. Traditional automated testing frameworks like Selenium, JUnit, and TestNG may notnative support visual recognition tasks, but they can be extended to validate object detection systems when integrated into custom test suites or machine learning pipelines [9]. For instance, TensorFlow's testing frameworks and PyTest can be utilized to automate the validation of object detection models, ensuring that these systems correctly identify and classify objects across diverse datasets [10]. Moreover, continuous integration tools such as Jenkins and Circle Continuous Integration (CircleCI) can be integrated into object detection workflows, automating the process of testing models after every iteration or code update [11]. These tools ensure that changes do not compromise the accuracy of the model, providing a scalable and efficient method to maintain high-performance object detection systems over time. Additionally, image-based automation tools like SikuliX, which use pattern matching for interface testing, could be adapted to validate visual components in applications that rely on object detection [12]. This approach helps confirm that objects are correctly rendered and identified by the system. However, automated testing for object detection presents its own challenges. Unlike traditional testing, which deals with defined inputs and outputs, object detection involves a degree of variability in visual data. Testing frameworks need to account for these complexities, including dynamic lighting, occlusions, and object distortions. While automated testing can help ensure consistency and speed, manual validation may still be required for edge cases. Overall, combining automated testing tools with custom scripts for object detection presents a promising way to streamline the testing of sophisticated machine learning models, ensuring accurate, reliable, and scalable performance [13].

The main objectives of these automated testing tools include improving testing efficiency, reducing manual effort, and ensuring comprehensive test coverage. The challenges primarily revolve around tool selection, cost, adaptability to diverse platforms, and integration with CI/CD pipelines. Additionally, the complexity of handling evolving technologies and the limitations of existing automation frameworks were notable issues. This comparative analysis underscores the importance of aligning tool capabilities with project needs, while addressing the inherent challenges of scalability, accuracy, and tool compatibility in different environments. The challenges, main aim and the objectives of the study are shown in table 1.

Table 1. Aim, objectives and challenges of the study

No	Main Aim	Objectives	Challenges
1	To compare various automated website testing tools.	Evaluate the effectiveness of automated testing tools	Full coverage of website quality with a single tool is difficult
2	Identify the most effective tool for assessing website quality	Compare tools based on their ability to provide comprehensive quality assessments	Limited time and budget make it challenging to test websites using multiple tools.
3	Evaluate tools based on performance, security, and reliability	Identify the most suitable automated testing tool for websites that can optimize testing time, resource usage, and cost.	Require extensive testing, which is difficult to manage within strict development cycles
4	Provide insights into the best tool for optimizing time, cost, and resources in small-scale websites	Assess the tools' ability to conduct tests efficiently	Difficult to Ensure comprehensive testing

The table 1 summarizes the main aim of comparing automated website testing tools to identify the most effective one for quality assessment. It highlights challenges such as limited tool coverage and constraints on time and budget. Additionally, it outlines objectives to evaluate tool performance and compare their effectiveness in ensuring efficient and comprehensive testing.

The rest of the paper is organized as follows: Section 2 describes the related work done on automated testing tools. The 3 section shows the widely used automated testing tools. The section 4 shows the discussion part of the paper. Finally, the paper concludes with the achieved result and future work regarding this project in 5.

2. Literature Review

Errors ignored in the early stage of software development may have a rippling effect, and may lead to time wasting and expensive to correct once it is deployed. So manual testing is normally carried in to practice for testing quality but some measurements shows that manual testing also requires great effort and time itself and is not reused for other software. Automation testing automates the steps of manual testing using automation tools. To overcome this problem, it is suggested to use automated testing tools for testing software quality [14]. The user interface of TestComplete is very straightforward and has efficient playback for testing so that to use scripts in future. TestComplete is also useful where data does need any security during testing process because it is it does not secure data in testing [15]. In 2019 authors Karthik & Jananisivapriya stated that Unified Functional Testing tool is suit-able when one is interested in data security even in testing too. The only disadvantage of the Unified Functional Testing (UFT) is that it is used commercially and one have to spend a high cost SoapUI is suitable for API testing as well as for non-functional testing such as performance and security test. It is free of cost tool. SoapUI Pro is capable of saving time so that it makes the testing fast and easy. The advanced version is also available at lower cost than TestComplete and UFT. Finally when concluded all results and features, the best tool among the four is SoapUI [16]. Wu et al., in 2021 focused on Practical Software Quality Techniques (PSQT) conference that traditional approaches such as manual testing may not be used for web testing in future such as for the WWW and e-commerce applications, besides all these positive aspects of automated testing tools, there exist few disadvantages too. When automated testing tools are not used, some projects suffocate under a stream of manual test scripts. When implemented on websites, testing tools management and the creation of automated testing scripts can consume all available time and resource [17]. Samli & Orman in 2023 utilizes a comparative analysis method to evaluate 14 web-based automated testing tools based on 20 distinct criteria. This method is chosen because it provides a comprehensive review of multiple tools, addressing gaps in previous studies that have only compared a few tools and criteria. The study focuses on important aspects such as cost, license, technical support, language and browser support, user experience, testing type, and hardware requirements. No direct experiments were conducted; instead, a detailed comparison of tool features was made using these criteria. The results offer an in-depth look at the advantages and limitations of each tool, helping testers and developers to select the most appropriate tool based on their specific needs. This study provides a valuable reference for those looking to optimize their tool selection for web-based automation testing [18]. Chaves et al., 2024, employs an automated testing method using the Robert Tool to evaluate mobile applications developed for Android at the Institute of Research and Development. This method was chosen for its ability to reduce manual testing efforts and increase efficiency. In the study conducted in 2023, the Robert Tool was used to automate 15 out of 35 test cases within a specific scope. The results indicated a 33% reduction in manual testing effort and an overall decrease of 714 hours in testing time. Survey feedback from the testing team highlighted the tool's ease of use and significant time savings. The accuracy of these results reflects a successful implementation of automation in mobile application testing, showcasing its practical benefits and efficiency improvements [19]. In another paper about automated testing tools authors Al-Khulaidi et al., 2024 proposes an evaluation framework for comparing and classifying automated testing tools used in agile software projects. This method was chosen to address the challenges of selecting appropriate tools amidst the variety of options available, considering factors such as programming language, system categorization, and tester expertise. The framework was developed through a literature review of agile testing methodologies and automation techniques. It evaluates tools based on key criteria like test design support, interfaces, and reporting capabilities. The study analyzed popular open-source and commercial tools, categorizing them by interface, code, design, and reporting features. The results offer guidelines for agile practitioners to select suitable tools, though specific accuracy metrics are not detailed, the framework aids in informed decision-making for tool adoption [20]. Haas et al., 2024 investigates the application of test optimization techniques from automated testing to manual testing processes. This method was chosen to explore how optimization strategies like test impact analysis and Pareto testing can enhance manual testing efficiency. The study analyzed five industrial cases across various domains, evaluating the costs and ben-efits of these techniques using data from 2,622 real-world failures. Results showed that optimized test suites, compared to manual ones, detected approximately 80% of failures while reducing execution time by 66%, versus 81% detection and 43% time saving for manual tests. The techniques also sped up time-to-first-failure by about 49 times. These findings indicate that automated test optimizations can significantly improve manual testing practices, though implementation requires addressing process-related limitations [21]. Aburas in 2024 employs a comparative analysis method to evaluate

16 widely used automated testing tools based on specific criteria. This method was chosen to simplify the selection process for software developers and testers by providing a structured comparison. The study involved organizing and comparing these tools based on their platform compatibility and testing functionality. The results offer a comparative view of the tools' features and applicability, aiding in the selection of appropriate testing tools. The study serves as a valuable reference for software developers, testers, and researchers, and supports educational purposes by offering insights into the effectiveness and suitability of various automated testing tools [22]. Garousi et al., 2024, employs a Multivocal Literature Review (MLR) to investigate AI-based test automation tools, which allows for a broad exploration of both academic and grey literature to gain a comprehensive understanding of the industry's current landscape. This method is used to analyze the available features, effectiveness, and efficiency of AI-powered testing tools. Additionally, an empirical assessment was conducted to evaluate two selected AI-based tools on two open-source software projects. By applying both AI-powered and non-AI approaches to the same feature, the study compares their performance and limitations. The results indicate that AI-based tools enhance testing efficiency and effectiveness, though some limitations remain. The study highlights the need for further improvement in AI-based testing tools to overcome current challenges and limitations [23].

Among the publications evaluated and the comparison shown in table 2, Garousi et al.'s (2024) [23] research provides the most extensive examination of AI-powered automated test-ing tools, examining 55 in total. This study not only evaluates academic literature but also conducts an empirical assessment of two chosen instruments, making it unique in its approach to combine theoretical and practical evaluation. Furthermore, the application of AI to improve testing speed and efficacy distinguishes this article from previous efforts that rely on traditional tools and approaches.

3. Widely Used Automated Testing Tools

Automated testing tools are software applications designed to execute pre-scripted tests on software applications automatically. They simulate user interactions and validate that the software behaves as expected, which helps in identifying bugs or inconsistencies early in the development process. These tools can run tests repeatedly and consistently, unlike manual testing, which can be time-consuming and error-prone. They support various types of testing, including functional, performance, and regression testing. This automation enhances test coverage, accelerates release cycles, and improves the overall quality of the software by ensuring that changes don't introduce new issues. Some of the widely used automated testing tools are shown in figure 1.

3.1. Selenium:

Selenium is an open-source tool designed for automating web browsers. It allows developers and testers to write scripts that simulate user interactions with web applications in multiple browsers [24]. Some of the advantages of selenium are, supports multiple programming languages (Java, C#, Python, etc.), Cross-browser testing across major browsers (Chrome, Firefox, Safari, etc.), Highly scalable for testing complex web applications. Automating a login process on a website across different browsers is one of the examples of selenium.

3.2. JUnit:

JUnit is a unit testing framework for Java applications. It helps developers write and run repeatable test cases for individual components or methods in Java code [25]. Junit Simplifies unit testing with annotations and assertions. Facilitates CI by integrating with CI tools like Jenkins. Offers fast feedback during development. Testing a method that calculates discounts in an e-commerce application is one of the examples of JUnit.

3.3. TestNG:

TestNG is a testing framework inspired by JUnit but with additional features for more complex testing needs. It supports different types of testing, such as unit, functional, end-to-end, and integration testing [26]. The advantage of TestNG is that it's Parallel test execution for faster testing. Built-in support for data-driven testing. Better configuration for handling complex test suites. Running parallel tests for a banking application's various modules.

3.4. Appium:

Appium is an open-source tool for automating mobile application testing. It enables automated testing of native, hybrid, and mobile web applications on iOS and Android platforms [27]. Appium supports cross-platform testing (iOS and Android). Supports multiple programming languages (Java, Ruby, Python, etc.). No need to modify the app code for testing. Automating the registration process in an Android shopping app is the example of Appium.

3.5. Jenkins:

Jenkins is an open-source automation server primarily used for CI/CD. It automates the process of building, testing, and deploying software projects [28]. Jenkins are Extensible with a wide range of plugins for

Table 2. Comparison of the related work done in field of automated testing tools

Focus Area	Tools Compared	Evaluation Criteria	Methodology	Results	Reference
Automated testing tools	Rational Functional Tester, QTP, Silk Test, Loadrunner	Usability, security, cost	Comparative analysis	Rational Functional Tester performs better in quality testing.	[14]
Automated performance testing	JMeter, TestComplete	Performance, efficiency, security	Performance analysis	JMeter excels in data security, TestComplete in efficiency.	[15]
Automated testing tools (API & non-functional)	Selenium, UFT	API testing, performance, cost	Comparative analysis	SoapUI offers best balance of cost and functionality.	[16]
Web testing using automated tools	-	Script creation, time, management	Case study	Time-consuming manual testing can be replaced with automation.	[17]
Web-based automated testing tools	14 web-based tools	Cost, license, language support, hardware requirements	Multi-criteria comparison	Provides comprehensive tool comparison across 20 dimensions.	[18]
Mobile application testing with Robert Tool	Robert Tool	Time efficiency, ease of use	Empirical evaluation	33% reduction in manual testing efforts.	[19]
Agile software testing tools	-	Test design support, interfaces, reporting	Literature review, framework	Framework guides agile practitioners in tool selection.	[20]
Optimization techniques for testing	-	Test execution time, failure detection	Historical analysis, survey	66% reduction in execution time, 80% failure detection rate.	[21]
Platform compatibility and testing tools	16 automated testing tools	Compatibility, functionality	Comparative analysis	Simplified tool selection based on platform compatibility.	[22]
AI-powered test automation tools	55 AI-based tools	Features, efficiency, effectiveness	Multivocal Literature Review (MLR)	AI tools enhance efficiency but have limitations.	[23]

integrating with various testing tools. Facilitates automated testing and reporting after each build. Highly customizable to fit different CI workflows. Automating test runs for each code commit in a web application.

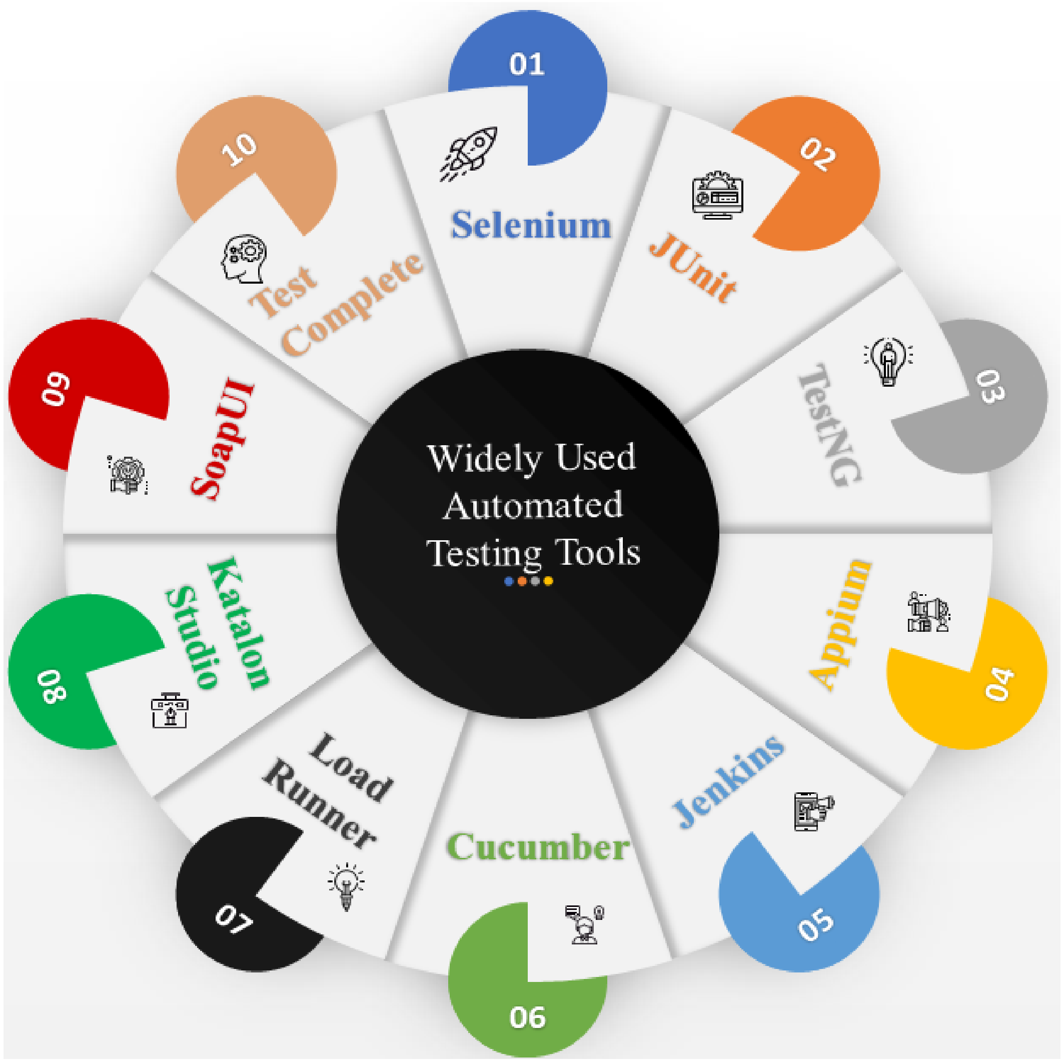

Figure 1. The 10 most widely used automated testing tools

3.6. Cucumber:

Cucumber is a tool that supports behavior-driven development (BDD). It allows tests to be written in plain language (like English) that non-technical stakeholders can understand [29]. The advantage of cucumber is that it bridges the gap between developers, testers, and business stakeholders. Supports multiple programming languages (Java, Ruby, etc.). Promotes collaboration in teams by using user stories in tests. Writing user acceptance tests for a feature in an e-commerce platform using plain language is the example of cucumber.

3.7. LoadRunner:

LoadRunner is a performance testing tool. It simulates virtual users to test how an application behaves under load [30]. LoadRunner Can test web and mobile applications for performance bottlenecks. Supports a wide range of protocols (HTTP, SOAP, WebSocket, etc.). Provides detailed performance metrics for optimization. One of the example of LoadRunner is the testing the server load handling capacity of an online banking application.

3.8. Katalon Studio

Katalon Studio is an all-in-one automated testing tool. It provides a user-friendly platform for testing web, mobile, API, and desktop applications [31]. Katalon Studio does not require advanced coding skills. Supports both record-and-playback and scripting for tests. Integrates with CI/CD tools like Jenkins and Git. The example of Katalon Studio is automating login and transaction validation in a mobile banking app.

3.9. SoapUI

SoapUI is a tool for testing APIs, particularly SOAP and REST services. It allows developers to automate functional, performance, and security tests for APIs [32]. The advantage of SoapUI is that it supports complex API testing scenarios. Includes features for load testing and security testing. User-friendly interface for creating test cases. Testing the response time and accuracy of a RESTful API used in an inventory management system.

3.10. TestComplete:

TestComplete is a commercial automated testing tool. It enables automated testing for desktop, web, and mobile applications [33]. TestComplete Supports multiple scripting languages (JavaScript, Python, VBScript, etc.). Can automate GUI testing across different platforms. Integrates well with other CI/CD tools for a seamless workflow. Example of TestComplete is the automating the testing of a desktop accounting software's user interface.

The comparison of each automated testing tool method is shown in table 3. From table 3 we can see for most projects that require a balance between web, mobile, and functional testing, TestComplete or Katalon Studio are the most versatile and user-friendly. However, for teams focused purely on web testing or with specific needs (e.g., API testing or performance testing), specialized tools like Selenium, SoapUI, or LoadRunner may be the better choice.

4. Disscusion

In the pursuit of delivering high-quality software, the selection of an appropriate testing tool is a critical decision for both developers and testers. This study aimed to explore various automated testing tools that are widely used in the industry and to compare them based on their functionalities, platform support, ease of use, and suitability for different types of testing. The analysis reveals that no single tool is universally superior, as each tool offers unique strengths depending on the specific requirements of a project. One of the key observations from the comparison is that tools like Selenium and Appium remain the go-to choices for web and mo-bile testing, respectively [34]. Their open-source nature, combined with a large community of users, makes them highly flexible and adaptable to different project environments. However, while these tools excel in their respective domains, they require a certain level of expertise to set up and configure, which might be a challenge for teams with limited technical knowledge.
On the other hand, more comprehensive tools such as TestComplete and Katalon Studio offer a user-friendly approach to testing across multiple platforms (web, mobile, and desktop) [35]. These tools simplify the testing process by providing built-in features that require minimal coding skills, which is a significant advantage for teams aiming to streamline their testing efforts without compromising on quality. In particular, Katalon Studio stands out as a versatile option for teams that require a balance between functionality and ease of use, making it suitable for smaller teams or those less familiar with scripting. Performance testing remains an essential component in ensuring that software can handle expected loads, and LoadRunner continues to dominate this space. Its ability to simulate high volumes of users and provide detailed performance metrics is unmatched, although it requires specialised knowledge to fully utilise its capabilities.

SoapUI plays a similar role in the API testing domain, where it simplifies the process of testing Representational State Transfer (REST) and SOAP services with minimal configuration required. One common factor across all these tools is the increasing integration with CI/CD pipelines, with Jenkins being the leader in automating testing processes. As software development shifts toward more frequent releases, the importance of seamless integration between testing tools and CI systems cannot be overstated. Tools like Selenium and Appium are highly compatible with CI systems like Jenkins, ensuring that automated tests can be run as part of the build process, providing faster feedback and reducing time-to-market. However, the discussion would be incomplete without acknowledging the limitations that come with automation tools. While automation reduces manual effort and accelerates testing cycles, it is not without its challenges. For example, automating certain complex scenarios or highly dynamic web elements can be difficult, requiring continuous maintenance of test scripts.

The figure 2 shows the automated testing tools ease of use, the platform that these testing tools support and their main key strengths.

Moreover, while tools like TestNG and JUnit excel in unit testing, they are not well-suited for more complex end-to-end testing scenarios without additional configurations or integration's. In addition, while commercial tools like TestComplete and LoadRunner offer extensive features, their cost can be prohibitive for smaller teams or startups. Open-source alternatives like Selenium and Appium provide a cost-effective solution, but they often require more setup and troubleshooting, which can offset some of the savings in terms of time and resources. Finally, there is the issue of scalability. As projects grow and become more complex, the scalability of the chosen tool becomes increasingly important [36]. Tools that

Table 3. Comparison of the widely used automated testing tools

Tool	Purpose	Ease of Use	Language Support	Platform Support	Types of Testing	Key Strengths
Selenium	Web application testing	Moderate	Multiple (Java, C#, Python, etc.)	Web (Cross-browser)	Functional	Best for cross-browser web automation
JUnit	Unit Testing for Java	Easy	Java	Java applications	Unit	Simple unit testing in Java
TestNG	Advanced testing for Java	Moderate	Java	Java applications	Unit, Functional	Supports parallel and data-driven testing
Appium	Mobile app testing	Moderate	Multiple (Java, Python, etc.)	iOS, Android	Functional, Mobile	Best for cross-platform mobile testing
Jenkins	CI/CD with testing integration	Easy	Multiple (Via plugins)	Cross-platforms	CI/CD, Functional	Excellent for automating testing in CI workflows
Cucumber	Behavior-driven development (BDD)	Easy	Multiple (Java, Ruby, etc.)	Cross-platform	Functional	Best for BDD and bridging tech/non-tech collaboration
Load-Runner	Performance Testing	Complex	Multiple	Web, Mobile	Performance	Industry-leading for load and performance testing
Katalon Studio	All-in-one testing tool	Easy	Groovy, Java, etc.	Web, Mobile, API, Desktop	Functional, API	User-friendly, minimal coding required
SoapUI	API testing	Easy	Groovy, Java	Web (API testing)	API, Performance	Best for SOAP/REST API testing
Test-Complete	Comprehensive automation	Moderate	Multiple (JavaScript, Python, etc.)	Web, Mobile, Desktop	Functional	Broad platform support for GUI testing

support parallel execution and integration with cloud-based testing services are better equipped to handle large-scale testing. In this regard, tools like TestNG and Katalon Studio, with their support for parallel and data-driven testing, provide a significant advantage in larger environments.

5. Conclusion and Future Work

The comparative analysis of automated testing tools demonstrates that no single tool is universally optimal for all projects. The decision on which tool to adopt should be based on the specific requirements of the software project, such as the platform being tested, the complexity of the test scenarios, the technical expertise of the testing team, and the budget available. Open-source tools like Selenium and Appium offer flexibility and cost-effectiveness for web and mobile testing, respectively, while comprehensive solutions like TestComplete and Katalon Studio provide greater ease of use and broader platform support, making them suitable for less technical teams. In specialized areas like performance and API testing, LoadRunner and SoapUI remain strong choices, though they may present challenges in terms of cost and complexity.

Despite the numerous benefits these tools offer, there are notable limitations to address. Automation, while reducing manual effort and speeding up test cycles, still requires significant maintenance and technical proficiency, particularly for complex scenarios. Additionally, cost considerations and the scalability of tools become critical as projects expand in size and complexity. Therefore, ongoing assessment and adaptation of testing strategies are essential to ensure that the chosen tools continue to meet the evolving demands of software development.

Looking ahead, there are several areas for future research and development in the realm of automated testing tools. First, greater focus on artificial intelligence and machine learning in test automation could improve the ability to predict and prioritize test cases, further reducing the manual effort required for

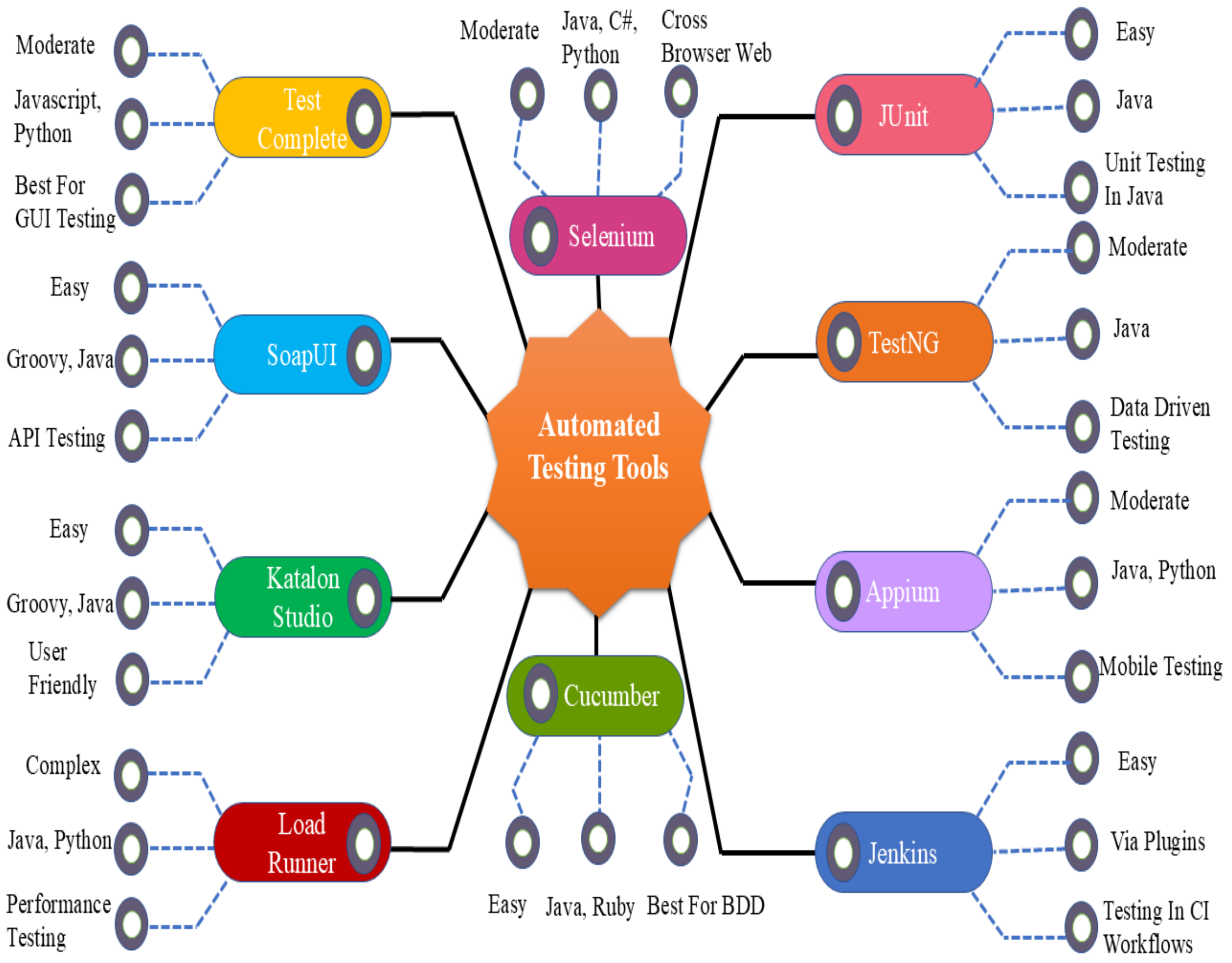

Figure 2. Automated testing tools ease of use, platform support and key strengths

test suite maintenance. The integration of AI-driven optimization techniques, such as self-healing tests, could minimize the impact of frequent code changes on test stability.

Additionally, cloud-based testing platforms offer promising solutions for scaling testing processes, enabling parallel execution across a vast number of environments. Future work could explore the development of more robust and cost-effective cloud integration's, allowing for better scalability, particularly for large-scale projects.

Another area for future exploration is the improvement of collaboration between development and testing teams, through more advanced behavior-driven development (BDD) tools like Cucumber. By fostering a deeper alignment between technical and non-technical stakeholders, such tools can bridge communication gaps and ensure that both parties have a clear understanding of testing objectives. Finally, as CI/CD pipelines become the norm in modern software development, there is a need for testing tools to enhance their integration capabilities. Future efforts could focus on automating not just test execution but also test result analysis and feedback loops, reducing the time-to-feedback and further streamlining the software delivery process.

References

[1] Homès, B. (2024). Fundamentals of software testing. John Wiley & Sons.

[2] Wang, J., Huang, Y., Chen, C., Liu, Z., Wang, S., & Wang, Q. (2024). Software test-ing with large language models: Survey, landscape, and vision. IEEE Transactions on Software Engineering.

[3] Alshazly, A. A., Elfatatry, A. M., & Abougabal, M. S. (2014). Detecting defects in software requirements specification. Alexandria Engineering Journal, 53(3), 513-527.

[4] Xu, G., Khan, A. S., Moshayedi, A. J., Zhang, X., & Shuxin, Y. (2022). The object detection, perspective and obstacles in robotic: a review. EAI Endorsed Transactions on AI and Robotics, 1(1).

[5] Latifi Rostami, S. A., Ghoddosian, A., Kolahdooz, A., & Zhang, J. (2022). Topology optimization of continuum structures under geometric uncertainty using a new extended finite element method. Engineering Optimization, 54(10), 1692-1708.

[6] Yu, X., Liu, L., Hu, X., Keung, J., Xia, X., & Lo, D. (2024, September). Practitioners' Expectations on Automated Test Generation. In Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (pp. 1618-1630).

[7] Kolahdooz, A., Nourouzi, S., Bakhshi Jooybari, M., & Hosseinipour, S. J. (2014). Experimental investigation of thixoforging parameters effects on the microstructure and mechanical properties of the helical gearbox cap. Journal of Mechanical Science and Technology, 28, 4257-4265.

[8] Moshayedi, A. J., Khan, A. S., Yang, S., & Zanjani, S. M. (2022, April). Personal image classifier based handy pipe defect recognizer (hpd): Design and test. In 2022 7th International Conference on Intelligent Computing and Signal Processing (ICSP) (pp. 1721-1728). IEEE.

[9] Hanna, M., Aboutabl, A. E., & Mostafa, M. S. M. (2018). Automated software testing framework for web applications. International Journal of Applied Engineering Re-search, 13(11), 9758-9767.

[10] Zarei, M., Moshayedi, A. J., Zhong, Y., Khan, A. S., Kolahdooz, A., & Andani, M. E. (2023, January). Indoor UAV object detection algorithms on three processors: implementation test and comparison. In 2023 3rd International Conference on Consumer Electronics and Computer Engineering (ICCECE) (pp. 812-819). IEEE.

[11] Camacho, N. G. (2024). Unlocking the Potential of AI/ML in DevSecOps: Effective Strategies and Optimal Practices. Journal of Artificial Intelligence General science (JAIGS) ISSN: 3006-4023, 3(1), 106-115.

[12] Moshayedi, A. J., Uddin, N. M. I., Khan, A. S., Zhu, J., & Emadi Andani, M. (2023). Designing and Developing a Vision-Based System to Investigate the Emotional Ef-fects of News on Short Sleep at Noon: An Experimental Case Study. Sensors, 23(20), 8422.

[13] Amit, Y., Felzenszwalb, P., & Girshick, R. (2021). Object detection. In Computer Vi-sion: A Reference Guide (pp. 875-883). Cham: Springer International Publishing.

[14] R. N. Khan and S. Gupta, "Comparative Study of Automated Testing Tools: Rational Functional Tester, Quick Test Professional, Silk Test and Loadrunner," Int. J. Adv. Technol. Eng. Sci., vol. 3, no. 01, pp. 167–172, 2015

[15] R. K. Lenka, S. Mamgain, S. Kumar, and R. K. Barik, "Performance analysis of au-tomated testing tools: JMeter and TestComplete," in 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), IEEE, 2018, pp. 399–407.

[16] M. Karthik and V. K. Jananisivapriya, "Comparison of Software Test Automation Tools-Selenium and UFT," Am. Int. J. Res. Formal, Appl. Nat. Sci., p. 5, 2019.

[17] H. Wu et al., "Peculiar: Smart contract vulnerability detection based on crucial data flow graph and pre-training techniques," in 2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE), IEEE, 2021, pp. 378–389.

[18] 18. Samlı, R., & Orman, Z. (2023). A comprehensive overview of web-based automated testing tools. İleri Mühendislik Çalışmaları ve Teknolojileri Dergisi, 4(1), 13-28.

[19] Chaves, L. C., Oliveira, F. C. M., Tiago, L. A., & Castro, R. G. V. (2024, May). Robert: An Automated Tool to Perform Mobile Application Test. In Proceedings of the 2024 10th International Conference on Computer Technology Applications (pp. 33-36).

[20] Moseh, M. A., Al-Khulaidi, N. A., Gumaei, A. H., Alsabry, A., & Musleh, A. A. (2024, August). Classification and Evaluation Framework of Automated testing tools for agile software: Technical Review. In 2024 4th International Conference on Emerging Smart Technologies and Applications (eSmarTA) (pp. 1-12). IEEE.

[21] Haas, R., Nömmer, R., Juergens, E., & Apel, S. (2024). Optimization of Automated and Manual Software Tests in Industrial Practice: A Survey and Historical Analy-sis. IEEE Transactions on Software Engineering.

[22] Aburas, A. (2024, May). Choosing the Right Automated Software Testing Tools. In 2024 IEEE 4th International Maghreb Meeting of the Conference on Sciences and Techniques of Automatic Control and Computer Engineering (MI-STA) (pp. 31-35). IEEE.

[23] Garousi, V., Joy, N., & Keleş, A. B. (2024). AI-powered test automation tools: A sys-tematic review and empirical evaluation. arXiv preprint arXiv:2409.00411.

[24] Jha, N., Popli, R., Chakraborty, S., & Kumar, P. (2022). Software Test Automation Using Selenium and Machine Learning. In Proceedings of First International Conference on Computational Electronics for Wireless Communications: ICCWC 2021 (pp. 419-429). Springer Singapore.

[25] Abdullin, A., & Akhin, M. (2024, April). Kex at the SBFT 2024 Tool Competition. In Proceedings of the 17th ACM/IEEE International Workshop on Search-Based and Fuzz Testing (pp. 65-66).

[26] Prasad, L., Yadav, R., & Vore, N. (2021). A systematic literature review of automated software testing tool. In Proceedings of 3rd International Conference on Computing Informatics and Networks: ICCIN 2020 (pp. 101-123). Springer Singapore.

[27] Moshayedi, A. J., Roy, A. S., Ghorbani, H., Lotfi, H., Zhang, X., & Liao, L. (2024, May 9). A novel IoT-enabled portable, secure automatic self-lecture attendance system: Design, development and comparison. International Journal of Electronic Securi-ty and Digital Forensics.

[28] Ma, X. (2024). Development and Automation of a Web Applications Using FastAPI, Jenkins, and Robot Framework.

[29] Mughal, A. H. (2024). Advancing BDD Software Testing: Dynamic Scenario Re-Usability And Step Auto-Complete For Cucumber Framework. arXiv preprint arXiv:2402.15928.

[30] Moshayedi, A. J., Roy, A. S., Liao, L., Lan, H., Gheisari, M., Abbasi, A., & Bamakan, S. M. (2021). Automation attendance systems approaches: a practical review. BOHR Int. J. Internet Things Artif. Intell. Mach. Learn, 1, 23-31.

[31] Safaat, G., & Tjhin, V. U. (2024). Analysis of Quality Assurance Performance in the Application of Manual Testing and Automation Testing for Software Product Test-ing. Indonesian Interdisciplinary Journal of Sharia Economics (IIJSE), 7(2), 1987-1996.

[32] 32. Raj, V., & Varri, U. S. (2024). 13 An Automated Approach. Cloud of Things: Foun-dations, Applications, and Challenges, 223.

[33] Prasad, L., Yadav, R., & Vore, N. (2021). A systematic literature review of automated software testing tool. In Proceedings of 3rd International Conference on Computing Informatics and Networks: ICCIN 2020 (pp. 101-123). Springer Singapore.

[34] Dias, T., Batista, A., Maia, E., & Praça, I. (2023, July). TestLab: An Intelligent Automated Software Testing Framework. In International Symposium on Distributed Computing and Artificial Intelligence (pp. 355-364). Cham: Springer Nature Switzerland.

[35] Moshayedi, A. J., Soleimani, M., Marani, M., Yang, S., Razi, A., & Andani, M. E. (2023, June). Fingerprint Identification Banking (FIB); Affordable and Secure Bio-metric IOT Design. In 2023 4th International Seminar on Artificial Intelligence, Net-working and Information Technology (AINIT) (pp. 384-390). IEEE.

[36] 36. Melyawati, N. L. P., Asana, I. M. D. P., Putri, N. W. S., Atmaja, K. J., & Sudipa, I. G. I. (2024). Comparison of Automation Testing On Card Printer Project Using Play-wright And Selenium Tools. Journal of Computer Networks, Architecture and High Performance Computing, 6(3), 1309-1320.