IJCSC 0973-7391 Volume 7 • Number 2 March 2016 - Sept 2016 pp. 262-269 available online at www.csjournals.com

A Comparative Study of Automation Tools and Frameworks in Automated Software Testing

Er. Seema Rani
Assistant Professor, Computer Science and Engineering, CDLSIET, Sirsa.

ABSTRACT:
In response to the growing need for "Quality at Speed," or software produced with high standards, faster and more efficient software testing execution is required to ensure software quality. Testing software requires the use of appropriate testing methods and test automation tools/frameworks in order for the process to be successful and effective. To fully test software and ensure its quality, it is often essential to combine a few appropriate testing approaches. Similar to this, no single tool can satisfy every criteria for automated testing, thus making the process of choosing the superior tool more tough. Software testing can only be successful and efficient if the various testing techniques, tools, and frameworks are understood. This research provides a thorough examination of automation test frameworks and tools. Adequate utilization of automation software testing frameworks and tools is required to produce high-quality software that meets client requirements. Comprehensive standards are still necessary even if software testing and automated testing techniques have previously been the subject of much research. The testing teams will be able to choose the best option based on the requirements with the aid of this research. The review's findings indicate that, depending on the situation and budget, the most often cast-off tools are Selenium, UFT, Ranorex Studio, JMeter etc.The importance of subjects pertaining to automated web testing technologies is increasing in published research. Unfortunately, there isn't a unique approach or framework that can fully manage and satisfy all requirements for automated web testing
Keywords: Taxonomy; Survey; Software Testing; Framework; Automation; Tools.

I INTRODUCTION

The significance of software quality and management has also become more widely recognized as computer software usage has increased [1]. Because software is capable to impact millions of people in many ways, it has become an indispensable aspect of our everyday lives, necessitating the usage of secure and dependable software. Because human beings make mistakes, faults are an inevitable component of software development, according to the principles of human-centered software development. Software flaws have the potential to be fatal and have a detrimental impact on real-time performance. These kinds of mistakes get more expensive as development goes on, so it's critical to address them early. Software engineering includes software testing, which has to do with the final product's quality. In order to guarantee software quality, the test is used to confirm, validate, and assess the dependability of software products [2]. Terrible things could happen if the software system’s quality is compromised. Since the late 1970s, more personnel and time have been dedicated to the software testing process as a result of all these problems. As a result, due to the fact that software is used in everything from mission-critical systems to everyday life, software testing has grown to be one of the most challenging and necessary processes for businesses, organizations, and researchers [3]. Front-end and back-end testing are two examples of the different aspects that go into software testing [4]. Software testing techniques come in a wide variety, both functional and non-functional. System testing, acceptability testing, unit testing, and integration testing are among the approaches used in functional testing, and they should be used in that sequence. Various non-functional testing approaches, such as performance, security, usability, and compatibility testing, are associated with the functional features of software products. Software testing fundamentals include determining whether the chosen test methods and test types are appropriate for a given piece of software, identifying the software requirements, choosing a verifiable and validation technique, conducting a scope check, documenting test processes, and more [5]. Software testing can be carried out automatically or manually [6]. Human input, analysis, and evaluation are all necessary steps in the manual software testing process. Manual testing is inherently risky since they involve human interaction. Mainly because taking the test repeatedly wears individuals out most of the time. It takes a lot of effort and patience to develop test cases for the manual testing process, which verifies if the program is correct. The process of automating test tasks, such as creating test scenarios, running and verifying them, and using automated tools, is called automated software testing. It is possible to automate all of the tests or only some of the test cases in an automated software test. The majority of software firms employ automated software testing to get advantages including less human labor, faster time to market, and better quality. Tests must be completed faster and with less effort in order for automated software testing to be effective. This has to do with choosing "software automation test tool" that will be applied. The "software automation test tools" that are designed for automation testing exhibit distinct characteristics based on their respective applications. For example, some are used for testing websites, and some are for mobile applications. Moreover, the programming languages and test methodologies employed determine the differences between these test automation solutions.

Over the years, numerous automated testing tools have been developed. Nonetheless, consumers find it challenging to select the appropriate test instrument in this scenario. Numerous researches that analyze and assess automation testing techniques may be found in the literature [7]. Despite their great value, these studies doesn’t address factors like code requirements, performance analysis, test levels, and interfaces used. This study is motivated by the desire to fill this research gap. The study's primary goal is to present a comparative analysis of significant and well-liked test automation solutions. Its goals are to provide light on upcoming research and to provide an overview of the automation technologies and general framework that should be taken into account while designing test processes. Selecting a testing tool is a crucial stage in the automation testing process. This study offers a variety of data and inputs to be taken into account during these elections. Thus, this effort will contribute to the body of knowledge in this field. and be very helpful to software and IT professionals as well as academic research scholars. This is how this research article is organized: The first part is an introduction; the second part discusses automated software testing and its categories; the third part discusses test automation frameworks and their various kinds. A few well-known test automation tools were introduced and contrasted in Part IV, and the article is concluded in Part V.

II. RELATED WORKS

For integrated functional and security testing of software systems, an automated test generating technique is described [8]. Automating test generation and execution is a highly desirable way to increase testing efficiency and lower costs. Repeatable tests and more frequent test runs made possible by automation allow for additional test cycles. A framework for automated testing is proposed, utilizing TestNG and Selenium Web driver, and its specifics are discussed. Testers may quickly and effectively write their test cases with this framework [9]. Currently, most inspection and testing procedures are carried out independently. One area of research that shows promise for utilizing additional synergistic effects is the combination of inspection and testing approaches [10]. Authors exemplifies the taxonomy of several automated testing technologies, including load, management, and functional testing [11]. Adopting the web for software product development has several benefits, the primary ones being. Cross-platform access from any device with an Internet connection, free feature upgrades for each customer, free installation, and independence from client operating systems [12]. Undertook a comparison analysis of automated testing technologies, including Selenium, Watir, Sahi etc., based on parameters including test script generation effort, script playback capability, result and price. The usage of open source automated testing technologies has significantly reduced the testing process's total cost as well as its execution time. This study's main goal is to evaluate and compare the features and capabilities of the testing tools enabling a user to choose the one that best suits their needs for a particular work. According to the poll, test automation offers advantages in terms of test coverage, reusability, and repeatability as well as reduced work required to execute tests. High initial costs for automation setup, tool selection, and training were a barrier [13]. This article addresses the necessity of automated testing during the software development process to produce software that is resilient, dependable, and of high quality. Based on the Selenium and J Meter, the authors have created an automated software testing framework for web applications [14]. To find out about present practices and areas for tool and process improvements in software testing, the authors surveyed enterprises and software testing professionals [15]. The aim is to figure out how to identify a variety of cutting-edge testing techniques and tools for software systems in order to guarantee that the produced product meets quality standards. In order to select a testing tool and technique that will save time and money, we will compare several testing tools based on the body of available literature. We will also compare various automated testing methodologies. To help users or developers choose the optimal online testing tool for their purposes, this study piece provides a feasibility assessment for both commercial and free source web testing tools [16]. One of the most important and difficult tasks in software engineering is figuring out which software testing tools are best for the project at hand.

III AUTOMATED SOFTWARE TESTING

Automated testing is the process of monitoring test execution and comparing expected and actual outcomes using software that is not part of the program being tested. Some manual testing processes are automated, but not all of them are, thanks to automation technologies [17]. Automated testing generally saves time. It makes it possible for the tester to finish a lot of tests quickly, including testing and other crucial and repetitive tasks that would be challenging to complete by hand. Test automation not only saves time but also money, effort, and quality of test tasks. It also helps to increase software accuracy. To create test cases and carry out testing, an experienced tester who is knowledgeable about automation tools and the software being tested is needed for test automation. Following Table 1 explains the advantages and disadvantages of automated testing.

Table 1: Automated Testing Advantages and Disadvantages

Advantages	Disadvantages
Superior to manual testing in terms of accuracy and speed of error detection.	Selecting the appropriate instrument takes a great deal of time, effort, and development planning.
Makes testing more effective by saving time and effort.	It needs familiarity with the testing tool
Increases test coverage because it's possible to utilize numerous test tools at once, allowing for parallel testing of various test scenarios.	The test is somewhat pricey and costly.
Repeatable automation test script.	Writing an automated test requires a certain level of qualification.

IV AUTOMATION TESTING FRAMEWORKS

Testing frameworks for automated testing tools come in a variety of forms. This section provides a quick explanation of a few of these techniques.

1. Framework for Modular Testing

Concepts from object-oriented programming serve as the foundation for the modular testing framework. With the help of this framework, the entire application being tested is divided into a number of coherent, independent modules. A unique and independent test script is required for each sub-module. Consequently, merging multiple test scripts yields a larger test script that encompasses many modules. These modules are isolated from other parts of the application by an abstraction layer [18].

2. Framework for Data-Driven Testing

The same functionality can be checked using other data sets in the application tests. Giving test data inside the automation code is therefore not a sensible choice. A more sensible and practical approach is to store data in other databases. This framework offers a way to separate these kinds of information. The data is kept in separate files, which results in separated test data. Traditionally, data is kept in pairs called "Key / Value."

3. A Framework for Keyword-Driven Testing

The keyword-based test framework allows it to store a specific set of code from the test script to an external data source, in addition to segregating test data from scripts. This code set determines keywords, which is why the framework has this name.

Examine Keywords and data are organized like a table. Regardless of the automation tool being chosen, keywords and test data must be assets [19].

4. Framework for Hybrid Testing

A mixture of several frames makes up the hybrid test frame. The fact that this installation makes use of various frameworks is its best feature. It is imperative to put up an adaptable framework for automated testing as more teams adopt an agile methodology. A hybrid framework is easier to modify in order to achieve optimal test outcomes.

5. A Framework for Behavior-Driven Testing

Automation is made possible by the Behavior Oriented Development framework in a way that is easy for testers to read and comprehend. The user does not need to be proficient in programming to use these frameworks.

It takes careful preparation and labor to evaluate and choose a framework. In order to achieve successful testing, it is imperative that when selecting a framework, the appropriate test automation tools be taken into account. Additionally, the framework should be able to adapt to changes in the software being tested as well as different automation tools.

V. PARAMETERS FOR TOOLS COMPARISON

Features for comparative analysis must be identified since evaluation can be conducted based on various automation tool specifications or attributes. Tools can be compared based on a variety of factors; as they serve the same objective, they will also have similar attributes. We selected the analytic criteria from a list of attributes that practitioners deemed significant when choosing test tools [20]. The table 2 below outlines the fundamental standards used to compare the various software testing solutions that are currently on the market.

Table 2: Criteria of Software Testing Tools

Criteria	Discription
Operating System Compatibility	Supported OS.
Browser Compatibility	Browser compatible.
Extendable	Extent, software can be expanded and given additional features.
Programming skills	Programming abilities needed.
Cost effectiveness	Licensed or Free.
Modifiable	The ability to be adjusted based on user needs.

VI. COMPARATIVE ANALYSIS OF AUTOMATION TESTING TOOLS

Table 3 presents a comparative analysis of various automated software testing methods selected based on factors.

Table 3: Comparative Analysis of Automation Software Testing Tools

Tool	Developer	Operating System Compatibility	Browser Compatibility	Language Supported	Extendable	Programming Skills	Cost	Modifiable
Selenium	Jason Huggins	Cross	Cross	Java Ruby, python, php C#, .net	Limited	Needed	Open source	Yes, modifiable
Test-Complete	Smart Bear	Windows	Chrome, Firefox, Opera, IE	Vb script, C#, jscript C++, delphi	Only to a limited degree	Needed	Market able	Restricted Modifiable
Watir	P. Rogers , P. Bret	Linux, Windows	Chrome, Firefox, Opera, IE, Netscape, Safari	Ruby	Limited	Partial	Open source	Yes, modifiable
Soap UI	SmartBear	Windows	Chrome, Firefox, IE	Java	Yes, extendible	Partial	Open source	Yes, modifiable
Load Runner	HP	Windows, Mac, Linux	Any Browser	C, Vb, Vbscript, C#, Javascript	Up to only certain extent	Partial, however scripts can be convoluted and challenging to read	Commercial	Limited
Silk Test	Microfocus	Windows	Cross Browser	VB.Net	Up to certain extent	Partial	Commercial	Supports but limited
Appvance	Appvance IQ	Cross platforms	Cross Browser	Java script	Yes, extendible	Partial	Commercial	Modifiable
Telerik Test Studio	Telerik	Windows Vista and Higher	All Browsers	VB.Net-C#	Not much extendible	Required	Commercial	Limited
Soap UI	SmartBear	Windows	Chrome, Firefox, IE	Java	Yes, extendible	Partial	Open source	Yes, modifiable
Rational Functional Tester	IBM	Windows, Linux	Chrome, Firefox, IE	Java, VB.net	Up to only certain extent	Required	Commercial	Limited
Apache-Jmeter	Apache	Cross platforms	No support for cross browser testing	Groovy, Java	Yes, extendible	Not required.	Open source	Yes, modifiable
Silk Test	Microfocus	Windows	Cross Browser	VB.net	Up to certain extent	Partial	Commercial	Supports but limited
Ranorex Studio	Ranorex GmbH	Windows	Chrome, Firefox, Opera, IE, Netscape, Safari	Vb script but supports .net, C++, C#, python	Not much extendible	Partial	Commercial	Limited
UFT	HP	Windows	Chrome, Firefox, IE	Vb script (supports java,.net, Delphi)	Up to only certain extent	Not required. Recommended for advanced test scripts	Commercial	Limited

VII CONCLUSION

Both well-known and obscure automated test tools are widely available on the market. These tools are available to completely fill in the gaps in the market and offer the consumer a variety of features. Examining several automated test tools and classifying their distinctive characteristics is the goal of this study. These tools cater to a variety of user types by showcasing both unique and common qualities. Unfortunately, many of the tools' features do not satisfy market demands on their own. Selenium is a popular choice among users in the market. This is primarily done to offer various selenium solutions for various applications. With its solutions, it offers automation for numerous projects serving various objectives. Because of this, it is impossible to discuss the merits of a single automated test tool. Numerous projects have a wide range of infrastructure issues that result from dynamics. In light of these issues and infrastructural conditions, several tools that provide various answers can be discussed. These tools are categorized and evaluated in this work based on their benefits and solutions..

REFERNCES

[1] Erdem, O. A. , Younis, A. "Yazılım Projelerinin Geliştirme Sürecinde Yönetim", Bilişim Teknolojileri Dergisi, 7(1), 1-9, 2014.

[2] Erdem, O. A. , Younis, A. E. "Yazılım Projelerinde Risk Yönetimi", Bilişim Teknolojileri Dergisi, 5(1), 1-6, 2012.

[3] I. Burnstein, Practical Software Testing: A Process-Oriented Approach, Springer Professional Computing, 2013.

[4] V.N. Nair, D. A. James, W. K. Ehrlich, et al, “A Statistical Assessment of Some Software Testing Strategies And Application Of Experimental Design Techniques”, Statistica Sinica, 8(1), 165–84, 1998.

[5] Grindal, Mats, Jeff Offutt, and Sten F. Andler. "Combination Testing Strategies:A Survey", Software Testing, Verification and Reliability, 15(3), 167-199, 2005.

[6] O. Taipale, J. Kasurinen, K. Karhu, et al, “Trade-Off Between Automated And Manual Software Testing”, Int. Journal of System Assurance Engineering and Management, 2(2), 114-125, 2011.

[7] M. Fewster and D. Graham, Software Test Automation: Effective Use of Test Execution Tools, ACM Press/Addison-Wesley Publishing Co., 1999.

[8] Xu, D., Xu, W., Kent, M., Thomas, L., & Wang, L., “An Automated Test Generation Technique for Software Quality Assurance. IEEE Transactions on Reliability, 64(1), 247–268. 2015. doi:10.1109/tr.2014.2354172.

[9] Satish Gojare, Rahul Joshi, Dhanashree Gaigaware, “Analysis and design of Selenium testing tool web driver automation testing framework,” published by Elsevier, ISBCC 15, pp. 341 346, 2015.

[10] Frank Elberzhager, Jürgen Münch, Vi Tran Ngoc Nha, “A systematic mapping study on the combination of static and dynamic quality assurance techniques”, Information and Software Technology, Volume 54, Issue 1, 2012, Pages 1-15, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2011.06.003.

[11] K Shaukat, “Taxonomy of automated software testing tool”, International Journal of Computer science and innovation, vol. 2015, pp. 7-18, 2015.

[12] S. Dogan, A. Betin-Can and V. Garousi, “Web application testing: A systematic literature review”, Journal of Systems and Software, vol.91, pp. 174-201, 2014.

[13] Dudekula Mohammad Rafi, Katam Reddy Kiran Moses, K. Petersen and M. V. Mäntylä, "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey", 7th International Workshop on Automation of Software Test (AST), Zurich, pp. 36-42, 2012.doi: 10.1109/IWAST.2012.6228988.

[14] F. Wang and W. Du, "A Test Automation Framework Based on WEB", IEEE/ACIS 11th International Conference on Computer and Information Science, Shanghai, pp. 683-687, 2012. doi: 10.1109/ICIS.2012.21.

[15] J. Lee, S. Kang and D. Lee, "Survey on software testing practices", in IET Software, vol. 6, no. 3, pp. 275-282, June 2012. doi: 10.1049/iet-sen.2011.0066.

[16] Monier, Mohamed & El-mahdy, Mahmoud., “Evaluation of automated web testing tools”, International Journal of Computer Applications Technology and Research. Vol 4. Pp 405-408, 2015. 10.7753/IJCATR0405.1014.

[17] Fernandes, J. D., & Fonzo, A. D., “When to Automate Your Testing (and When Not To)” , March 2013.

[18] Singh, I., & Tarika, B., “Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli and Watir”, International Journal of Information & Computation Technology, 4 (vol. 15), pp. 1507–1518, 2014.

[19] A. Bhargava, “Designing and implementing test automation frameworks with QTP : learn how to design and implement a test automation framework block by block”, Packt Publishing, 2013.

[20] Gorton, Ian. (2011), “Essential Software Architecture”, second edition, Springer, pp. 23-38, 2011. DOI 10.1007/3-540-28714-0.