Table of Contents

Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete

This article starts by defining what testing is and explaining that good testing should find all the defects existing in their system. The goal of this paper is to test and study Selenium 2.0.0 while comparing it with Quick Test Pro (QTP) and TestComplete. Unlike the other two options, Selenium is a free, open-source tool that offers the most simple setup, but still works just as well. QTP, as it is a paid tool, allows testers to work with a detailed GUI and can test whole java applets alongside a variety of different web applications. TestComplete supports various test types such as unit, functional and GUI, regression, and finally, distributed testing. Their main approach to determine which tool was better was to compare them based on these topics: Recording efficiency, Capability of generation of scripts, Data-driven testing, Reusability, Execution speed, Playback of the scripts, Easy to learn, and Cost. They chose these topics because they knew that each tool offered very similar features but each had its own advantages and disadvantages. Selenium offered a decent level of application support (primarily web applications and plug-in support), the highest amount of programming languages available, works on all OS platforms, requires significant knowledge of programming, does not work great with databases and the report generation was moderate at best compared to the other two. Ultimately, QTP was the best choice overall. This paper was not very useful because it did not demonstrate any real figures or stats that would help us implement this in practice. It was helpful as an introduction to the SDLC and the various testing tools that exist. While I would not consider it a critical reading for our chosen topic, I would say it is a good read to help introduce us to web application testing.

Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli and Watir

This paper is similar to the previous in the sense that they are comparing Selenium with two other web application testing tools, these tools being Sikuli and Watir. The problem they are trying to solve in this paper is to determine an answer to two primary questions: how can one differentiate between two tools on the basis of common features? Which open source testing tool is better in general? Sikuli uses images to generate test cases and has a very clear workflow to get it to work. Watir is an open-source Ruby tool with various variations. Like the last paper, this one also has its own testing criteria to evaluate each tool’s performance, they are recording capabilities, execution speed, scripts generation, data-driven testing, ease of learning, testing reports/output, supplement features. Their methodology to test these areas was to provide a clear definition of what each one was and then compare them to how each tool performed.

They found that Watir had the fastest execution time (ran a test to log into a google account), Selenium had the best overall recording capabilities and scripts generation and data-driven testing scores. Sikuli resulted in the best at ease of learning and tied with Selenium for testing reports. Finally, they concluded that Selenium was the best choice of the three because of its enhanced recording features, DDT, and ease of learning plus overall performance. I would say this paper was useful in practice and theory because unlike the first paper, they displayed their methods clearly and with supporting figures, graphs, and stats. I would classify this as a critical reading to our project because it helps us understand the pros and cons of Selenium.

Web Application Tests with Selenium

This paper starts by talking about Acceptance Testing and describing what it is and how it is useful. They give the example of how an email program's send button should only be active when there is at least one recipient selected. This paper focuses less on running tests on tools but rather explains in-depth how Selenium came to be and how it functions behind the scenes. They give a sample problem of how to test auto-completion of country codes using Selenium RC's option with Java. The article describes how Selenium offers four main versions for testers to utilize: Selenium Core, Selenium RC, Selenium Grid, and Selenium IDE. Core allows testers to use Selenese which is its native control language used to write test cases for web applications. This allows direct access to all of Selenium's commands with specified columns for each command. However, if the tester does not want to take the potentially long time to learn this language, they can use Selenium RC (remote control) which acts as a mask for Selenese. This means that users can use the same commands from Core, but write them using their preferred language such as Java or Python (and many more). Ultimately, I believe that this paper is not critical reading for our group to make our project better, but it would definitely help as supplementary reading. This is because this article describes the history of Selenium and what came before it, which in theory, helps us understand how it works more. It touches on enough topics to be helpful and broad, but at the same time, since they do not test any particular hypothesis/topic like the previous two papers, it only acts as an article to help explain what Selenium is (which is still helpful to some extent).

Automating functional tests using Selenium

This research paper describes the standard environment testing of a web application with Selenium, along with tools and frameworks such as FitNesse, CruiseControl, and Cobertura for the whole process of automating functional testing. The authors found that it is difficult to test web applications because of multi-tiered architecture, multiple

browsers, and web technologies. They want to create a methodology that can write satisfied tests before beginning the development of the application.

First, they write Selenium tests that are as independent and self-contained as possible since flexibility and maintainability are their priority. The team also wrote their own extension to make more readable tests. Then, they integrate Selenium tests into FitNesse, an application that includes the functions of an automated test tool, wiki, and web server. Users can make changes on a page and FitNesse keeps the old versions. Then, they used CruiseControl to add their Selenium tests to the continuous integration build. It can deploy the testing application and Selenium on the integration machines. Lastly, they use Cobertura to record the percentage of code accessed by tests.

They find a way to refactor the frontend code without fearing breaking the previously passing acceptance criteria for the application, but it is still possible to break an automated test in some circumstances. However, it is easy to demonstrate the Selenium tests and the test provides so much regression value that allows the developers to know the application is still stable. The paper has mentioned the whole process of automated function testing on a web application. However, the paper itself was published more than a decade ago. Selenium has more functions than the paper mentioned. Also, it does not cover back-ended applications. Therefore, it can only let us understand the basic concept of Selenium but not be a critical reading.

Agile testing with Selenium

This paper shows how the Selenium tool increases the speed of the test execution process and saves the cost of it by implementing agile testing. The authors suggest this software testing practice can be done by using automated scripts. That led to Selenium, an automated testing framework used to test web applications across platforms and different browsers. Selenium has one key feature: support for executing application tests on multiple browser platforms. It helps the developers understand the compatibility of their applications. The version that the authors use is Selenium 2, integration of WebDriver with Selenium Remote Control(Selenium1). The authors have developed their own framework called Pytest. The framework starts with combining all selenium 2.0 python test cases and runs them using nosetests against a variable of configurable parameters. (Browsers type, Selenium hub server, Selenium hub server Port, base URL, test report filename) After that, running all test cases in parallel per browser. Lastly, an XML-format JUnit report is produced. The research concludes that the benefits of automated testing are saving time and easy to maintain.

The paper mentions the steps of the testing process clearly. However, it is a lack of analysis of the tools that have been used. The finding is too bare-boned. There is no

sufficient material to support the conclusion. Moreover, the paper is outdated. Selenium has released version 3 but the paper is still using version 2. It is not useful in practice or theory. I will not include it as a critical reading for the topic.

Testing using selenium web driver

The objective of this article is to show the importance of software testing in the Software Development Life Cycle. The authors state that automated testing is better than manual testing because it saves time and has high accuracy. They use Selenium WebDriver to carry out test automation on a self-written application called “Lawyer’s Hub”. They have written test cases with different configurations such as browser types and login status. Maven is used to combine Selenium Web drivers for Java projects. After that, an XML file will be executed, and run all the test cases. The web driver will automatically fill in the data until there is a test case failure. The email report will represent the details of the testing result. A defect log will also record the id, the description, the priority, and the severity estimations of the defects. Overall, the finding is that automated testing is the best and easier approach to perform software testing, compared to manual testing. Those test cases can be reusable and the scripts are easy to maintain and document.

The article has included detailed instructions for software testing with Selenium Webdriver. It also has a brief description of the methodology and the tools used. The practice in the article is useful for our project since we will use similar tools and frameworks to build the project. Therefore, I will consider it as a critical reading.

SAHI vs. Selenium: A comparative analysis

This paper is a comparative analysis that looks at SAHI and Selenium as potential tools to test wireless based graphic user interfaces. The experiment uses the same machine to conduct tests using both tools and uses metrics such as time, efficiency, and execution speed. They looked at learnability as an important metric and provided support for their reasoning on why both tools are easy to learn. The conclusion states that automating your software testing process by using Selenium or SAHI increases effectiveness, efficiency, and testing coverage. Overall, this analysis was done quite well, the conclusion isn’t bias towards one tool or the other but the paper as a whole provides a solid breakdown of when one outshines the other. This paper is not very useful in theoretical computer science but is very practical if an one was deciding on the two tools and wanted an analysis on the difference between them.

Since our project is focusing on the use of selenium, it’s important to understand competing tools and how they differ. Therefore, I consider this a critical reading for this topic.

Data Driven Testing Framework using Selenium WebDriver

This paper aims to take a different approach to software testing by using a data driven approach using Selenium WebDriver and Excel. The paper details the benefits of automated testing over manual testing practices. The author has screenshots and shows code examples to demonstrate how this propose framework could work in the industry.

Overall, this paper is does not provide a convincing argument on how this new framework of testing is superior to existing methods. The paper is unprofessional because it is not written in an academic manner. It lacks grammatical correctness, uses inconsistent font sizes and references improperly. I do not consider this paper a critical reading for this topic.

Automation Testing Framework for Web Applications with Selenium WebDriver

This paper investigates the need of automation testing in the software development process. Selenium WebDriver is chosen as a means to create an automation testing framework to analyze the opportunities and threats of automation testing, specifically with Selenium WebDriver. The paper is more exploratory and does not focus on quantitative metrics such as speed, efficiency and reliability. The author explores different frameworks software testers could and their associated costs. The paper concludes that overall, Selenium's advantages greatly outweigh the risks involved in adopting a new tool. Selenium involves a great deal of work to setup, learn and scale but the consistency it provides is hard to find in traditional manual testing methodologies.

I think this paper should be considered a critical reading for this topic. It explores different ways the tool is used and where it falls short.

Wait, Wait. No, Tell Me. Analyzing Selenium Configuration Effects on Test Flakiness

The paper begins by providing an example of when non-deterministic (flaky) tests can cause great amounts of frustration and uncertainty for developers. Then, the authors outline their approach to analyzing configurations of Selenium using different underlying web browsers and timeout strategies in order to test stability and runtime performance. They provide examples

of when flakiness can occur such as a test case meant to verify the presence of a UI element before the browser has completely loaded, causing it to fail. In their work, they use the undergraduate software engineering class project iTrust2, an open-source project that is designed to provide students with an industry-like experience “by exposing them to a large system and continuous integration”. NC State University’s software undergraduate course uses iTrust2 as one of their primary tools. The motivation behind their work is to “proactively identify and remove test flakiness and improve performance in iTrust2by finding and implementing optimal Selenium and system configurations”. They came up with configurations of selenium that addressed 5 questions such as “what is the impact of the webDriver?”, “which wait methods are the most stable” and so on. The way they executed this was by first recording both the build after execution and a list of failing test cases. They then ran evaluations for each configuration on their procured systems and ultimately decided to use performance results to inform their selection of run configurations. Their results showed that there are implications for automated testing of web applications using Selenium with regards to flakiness and overall performance. They note that there is much potential for future works that consider explicit waits for page loading, hardware, and operating system improvements, and further implementation of WebDrivers.

DESIGN OF AUTOMATION SCRIPTS EXECUTION APPLICATION FOR SELENIUM WEBDRIVER AND TestNG FRAMEWORK

The article begins by introducing the reader to the concept of software testing in a straightforward and easy-to-follow manner. The authors clearly explain that in order to deliver software that is valid and of high quality, it must be tested manually or by using automation tools to identify defects. They dive a little deeper into automated tools pointing out that they help carry out activities like test design, execution, defect management, and overall coverage analysis in a fast and cost-effective way. Furthermore, they explain different software development models such as waterfall, agile, spiral, and the iterative enhancement model and then explain that they can be improved through automating steps. They further explain the nuances of manual testing and outline certain problems involved such as carrying out regression testing on a daily basis, and stipulate that automation could help overcome these problems. The authors compare three automation testing tools;, SELENIUM WebDriver2.0, QTP, and Test Complete. They highlight the differences between them such as Selenium’s support across operating systems such as Windows, Mac, and UNIX, while QTP only supports windows. They then dive even deeper into the specific features of each automation tool. They point out that selenium allows for designated test scripts to communicate with the browser directly. Then they give their proposed system of having a web project replace a stand-alone project that is run on an apache tomcat server. This has advantages such as keeping track of the number of test methods under execution and automatically summarizing the execution in a report. The end by giving a viability assessment and stating that the proposed system can be easily integrated with existing selenium and java projects that use TestNG frameworks

A Survey of the Selenium Ecosystem

The article presented by the authors provides a survey aimed to understand how the community uses Selenium and its ecosystem. They recognize the automation tool's popularity and seek to unravel the truth about how the public reacts when given seven categories which are "Selenium foundations, test development, system under test, test infrastructure, other frameworks, community, and personal experience". They begin by giving a background and supplying the reader with related work upfront in the pursuit of transparency. Afterward, the authors build confidence by outlining the design of the survey and highlighting its descriptive, evaluative, and formative nature. They received 72 completed questionnaires from 24 countries around the world including the US, India, Spain, United Kingdom, Canada, Japan, Argentina, and more. In the firsts category (foundations) they found that the preferred programming language used in Selenium scripts was Java, and PHP receiving the least votes. For the second category, they found that most people used chrome as their preferred test development, with Firefox coming in second. Third, they found that the vast majority of users developed web applications with Selenium. Fourth, the authors found nearly 60& of users test on their local browsers. As for the fifth category, they found JUnit was the preferred Java unit testing framework, and Pytest to be the preferred framework for python. As for the sixth category, they discovered online tutorials to be the greatest form of documentation for Selenium, and Stack Overflow to be the greatest source of support. Finally, they found that almost half of all users found Selenium to have problems with maintainability and flakiness. The article ends with a discussion about the results and concludes by mentioning the bright future of Selenium.

Analysis and Design of Selenium WebDriver Automation Testing Framework

This article analyzes a proposed automation testing framework that has been developed and implemented using selenium WebDriver. They introduce the article by giving a brief history and description of software testing and clearly highlighting the importance of developing reliable and correct tests. Then, the authors provide the reader with a background on Selenium and in an extremely easy-to-understand manner, along with its firefox plug-in Selenium IDE. After they ensure that the reader understands the nature of software testing, they provide their proposed work which is based on the selenium web driver and TestNG testing frameworks. The proposed system includes five components: Object Repository, Input Files, Utility Section, Test Suite, and Customization Test Reports. After declaring these five components, they go one-by-one and describe each component in detail. Then, they give examples of their system in action and provide results wherein their system had an overall pass rate of 95.42% as opposed to the 71.7% pass rate of traditional approaches. They then conclude the article by summarizing the new automated testing framework and insist that it is very useful for dynamically changing web applications.

References

1. H. Kaur and G. Gupta, “Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete,” *Int. Journal of Engineering Research and Applications*, vol. 3, no. 5, pp. 1739–1743, Sep. 2013, doi: 10.1.1.448.6743.
2. I. Singh and B. Tarika, “Comparative Analysis of Open Source Automated Software Testing Tools: Selenium, Sikuli and Watir,” *International Journal of Information & Computation Technology*, vol. 4, no. 15, pp. 1507–1518, 2014, doi: 10.13140/2.1.1418.4324.
3. A. Bruns, A. Kornstadt and D. Wichmann, "Web Application Tests with Selenium," in *IEEE Software*, vol. 26, no. 5, pp. 88-91, Sept.-Oct. 2009, doi: 10.1109/MS.2009.144.
4. A. Holmes and M. Kellogg, "Automating functional tests using Selenium," *AGILE 2006 (AGILE'06)*, Minneapolis, MN, 2006, pp. 6 pp.-275, doi: 10.1109/AGILE.2006.19.
5. R. A. Razak and F. R. Fahrurazi, "Agile testing with Selenium," *2011 Malaysian Conference in Software Engineering*, Johor Bahru, 2011, pp. 217-219, doi: 10.1109/MySEC.2011.6140672.
6. P. Ramya, V. Sindhura and P. V. Sagar, "Testing using selenium web driver," *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, Coimbatore, 2017, pp. 1-7, doi: 10.1109/ICECCT.2017.8117878.
7. T. J. Naidu, N. A. Basri and S. Nagenthram, "SAHI vs. Selenium: A comparative analysis," *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, Mysore, 2014, pp. 967-970, doi: 10.1109/IC3I.2014.7019594.
8. C. Chandraprabha, A. Kumar, and S. Saxena, “Data Driven Testing Framework using Selenium WebDriver,” *International Journal of Computer Applications*, vol. 118, no. 18, pp. 18–23, 2015.
9. E. Vila, G. Novakova, and D. Todorova, “Automation Testing Framework for Web Applications with Selenium WebDriver,” *Proceedings of the International Conference on Advances in Image Processing*, Aug. 2017, doi:10.5121/csit.2019.91803.
10. K. Presler-Marshall, E. Horton, S. Heckman and K. Stolee, "Wait, Wait. No, Tell Me. Analyzing Selenium Configuration Effects on Test Flakiness," *2019 IEEE/ACM 14th International Workshop on Automation of Software Test (AST)*, Montreal, QC, Canada, 2019, pp. 7-13, doi: 10.1109/AST.2019.000-1.
11. R. Jain and R. Kaluri, “DESIGN OF AUTOMATION SCRIPTS EXECUTION APPLICATION FOR SELENIUM WEBDRIVER AND TestNG FRAMEWORK,” *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 6, Apr. 2015.
12. B. García, M. Gallego, F. Gortázar, and M. Munoz-Organero, “A Survey of the Selenium Ecosystem,” *Electronics*, vol. 9, no. 7, p. 1067, Jun. 2020, doi: 10.3390/electronics9071067.

13. S. Gojare, R. Joshi, and D. Gaigaware, “Analysis and Design of Selenium WebDriver Automation Testing Framework,” *Procedia Computer Science*, vol. 50, pp. 341–346, 2015, doi: 10.1016/j.procs.2015.04.038.