International Journal of Engineering & Technology, 7 (3.7) (2018) 367-370

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Test automation tools: A comparison between Selenium, Jenkins and Codeception

***Ali Stouky, Btissam Jaoujane, Rachid Daoudi, Habiba Chaoui**

Laboratoire Genie des Systeme, l'equipe ADSI, ENSA de Kenitra, Morocco
**Corresponding author: *Ali Stouky: Laboratoire Genie des Systeme, l'equipe ADSI, ENSA de Kenitra, Morocco*
**Corresponding author E-mail: ali.stouky@gmail.com*

Abstract

Software development life cycle consists of many phases and software testing is one of them. It's a very important one and ensures the quality of the software by fixing bugs which can be done with automated testing to reduce human intervention and to save time and effort consumed in the manual testing. The entire process of testing can be automated to reduce time and cost easily with the help of automated testing tools. Thus, it is crucial to select the suitable tool that meet the needs of the software system. This paper provides a feasibility study for the most commonly used web testing tools to determine their usability and effectiveness.

Keywords: *Test Automation, software-testing tools, Functional testing.*

1. Introduction

Software testing can be defined as the process of checking if a system satisfies the specific requirements. The main goal behind the software testing process is to detect any existing anomalies in a software product [1]. It can be done manually or automatically; manual testing is error prone and a time-consuming process [2]. In other words, Automation Testing is the use of testing tools to cut the need of manual or human intervention, and discover defects manual testing cannot expose. In such cases, performing security checks and adding functional tests can be very tedious in the long-term and very costly, because it means requiring system administrators to spend a part of their time performing these security checks and to developers to add automated functional tests to each feature or recruiting new staff dedicated to these tasks. That is the part where automated testing comes to picture.
A testing automation framework is an execution environment for automated tests used to find problems at an earlier stage and solve them. Those testing tools are designed to generate automated tests and enhance the testing performance. Hence reducing time, cost and effort. All these tasks will be performed automatically.

2. Overview

Software Testing (ST): Software testing is the process of executing a program or system in order to find errors and defects in software as fast as possible, it represents any activity designed to evaluate the capability of a program or system and verifying that it successfully meets the requirements.
Nowadays, number of software system has been implemented as web-based applications. These web applications are very complex 2. It is exceptionally hard to test such complicated applications, not following a continuous approach will lead you to have longer periods between integrations and this will make it more difficult to find and fix problems.

Continuous Integration (CI): It is a practice that allows, as its name suggests, software engineers to deploy continuously the code sent after their commits. Each change is detected and triggers a series of automated tests before deploying to the shared repository, allowing teams to detect bugs early in order to have visibility and to gain Time, which is the more important reason why we should have a similar approach, the working group will Spend less time debugging and more time developing and adding new features. It makes the build self-testing and automated.
While in general automated testing is not strictly considered as a part of CI it stills typically imposed and it represents the weak point of the workflow. Software testing is a task that software engineers don't like to do and which consumes a lot of time and money over the life cycle. It consumes 40%-50% of total resources, 30% of total effort and 50%-60% of the total cost [3-4].

In recent years, Web applications have become increasingly popular and testing has become a key practice for developing quality software and follow a number of key principles. The quality of these web applications is one of the important factors while deploying these web applications [2]. And that is the part where testing plays a critical role to raise the quality of the software. In one hand, manual testing requires human intervention and it is time consuming, while on the other hand the workflow cycle becomes shorter and shorter as we progress in the project and this will make the software testing more difficult for the testing team, thus, automation testing came into picture to avoid all of the mentioned problems.

Functional Testing

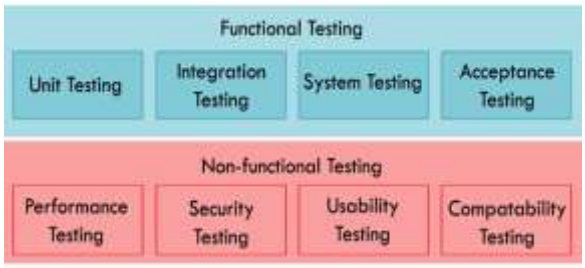


Figure 1: Functional and Non-Functional Testing.

By integrating and testing a project regularly we can detect bugs quickly, and locate defects more easily. Some Integration problems can cause a project to fail or even knock it off-schedule, that is why firms should consider adopting such technologies and they should also know that Automated Testing is cheap, while Not Testing Automatically is expensive, and Thus, Automation is a good way to reduce the cost and time. The test phase is a major challenge in software development and can be considered a fair opportunity that can significantly contribute to improving and optimizing the cost, quality and time to market of the software. This improvement came just in time when the software industries are struggling to cope with international competition by reducing their budgets and schedules [1]. Another important Approach to put under consideration, to gain more time and to fix bugs in a smart kind of way is Automate deployment, it presents the process of delivering software into production that have been passed the automated tests successfully. Additionally, Continuous Deployment and Continuous Integration are always related and by adopting them both developers not only cut down risks and fix bugs rapidly, but also move immediately to working software.

Figure 2: Integrating testing into the continuous integration and delivery process.

3. Literature Review

When automated testing came to picture, the companies not only wanted to test software adequately, but also as quickly and thoroughly as possible, cost, code quality, and time are the three big matters as we mentioned above.

All the developers that constitute the team usually work on a local version of the software on their own machines to avoid any problems that can affect other's work, so they add their modifications and the new features repeatedly, and then merge their new changes with the latest build of the project that is ready to be deployed.

For any version, there exists multiple in home local builds of that same version of code at each developer's site and while many developers are coding in the same time, it makes it hard to test every local version that a developer has modified or added, and as we know if the team works daily and most of the time nightly at home or weekly, we are going to have builds that are impossible to test adequately.

Thus, test automation is the right solution to deal with each functionality added. The question that comes to mind is What framework should we use, or how can we decide which solution is better for the company.

When automated testing came to picture, the companies not only wanted to test software adequately, but also as quickly and thoroughly as possible, cost, code quality, and time are the three. big matters as we mentioned above. In the past 10 years, many researches have been done in this scope. Amannejad in [5] presented how to automate the entire test process, from test case design, to test scripts, to test execution, and finally to evaluate test results. Being a step ahead Vahid Garousi. In V. Garousi and M. V Mäntylä [6] presented a multi-vocal review on test automation defining when to automate and what to automate. They gave in details the background of test automation. big matters as we mentioned above. In the past 10 years, many researches have been done in this scope.

Lately Mohamed Monier, Mahmoud Mohamed El-mahdy, focused on the Evalu-ation of automated web testing tools Computer. They provided a feasibility study for some few commercial and open source web-testing tools.

Divya Kumar [7] tried to identify and compare the tools in which the success of test automation depends on the three critical software dimensions of cost, time, and quality.

They gave some detailed statistics and calculations to enumerate the impact of test automation on a project, the results of their studies clearly show the positive effects that test automation may have on costs, quality and time to market of the software.

While in Yogesh Kumar [1] presented a comparative study on testing tools in or-der to do the selection of right automated software testing tool, they have discussed four tools that fit into their case study. In addition, Satish In S. Gojare, R. Joshi, and D. Gaigaware [8] have implemented an automation-testing framework for testing web applications using selenium WebDriver tool and mentioned that it is use-ful for dynamically changing web applications. Another these researches [9-11] they also have chosen to work with Selenium and that shows that it is commonly used for Functional Testing.

Table 1: Comparison of open source automated testing tools

Tools/ Features	Selenium	Codeception	Jenkins
1) Operating System/Platform	ALL	ALL	ALL
2) Browser Support	ALL Browsers	ALL Browsers	ALL Browsers
3) Language and Frameworks Support	Php, java,C#, javascript,perl, python,Ruby	Php,Ruby, Python, Java, .Net, Appium, NodeJS, Javascript, Robot Framework	Python, Ruby, Java, Android, C/C++
4) Ease Of Use	Needs a quite Expertise	Experience needed	Very easy to use
5) Type Tests	Unit, functional	Acceptance, functional, unit	Unit, Automated integration tests
6) Report Generation	Integration with jenkins can give good reporting & dashbord capabilitie	Several information is provided : execution time, statistics	Checkstyle, PHPMD, PHPCPD, HTMLReport…

To sum up, they mentioned Selenium as it is the most popular open-source tool for Web UI automation testing. A lot more researches have been done in this context witch highlight on the benefits and the value of test automation. However, no work, to the best to our knowledge, has been done to show the difference between the tools that we can use for continuous integration including all test phases and how can we use them.

Evaluation study : In our case study, we analyzed three test automation tools but decided to keep only two of them that best suit the development environment, which consists of a web application developed with PHP5 using Laravel: the PHP Framework for web artisans. Moreover, both database management systems: MySQL and MongoDB while we used the Bitbucket tool: a web-based hosting service for versions management.

In the following, we are going to present a detailed comparison between three of the most known open source tools for automation

testing frameworks in order to choose the tool that is adequate for our case study.

Jenkins: Jenkins helps automate software builds regularly. At each commit, it launches tests and informs developers if their code contains anomalies or not. These premature construction failures reduce costs, it is also a guarantee of robustness on the product software and allows teams to judge the quality of the software based on unbiased metrics..

In our case, it will be coupled with selenium, which will take care of functional testing. Furthermore, Jenkins can perform Continuous Integration natively, which is a good point, as it will help us in the whole process of testing.

Jenkins provides developers with the test results of each phase running. They only need to commit their changes and Jenkins will do the rest..

Table 2:. Comparison between "Before and After Jenkins"

Before Jenkins	After Jenkins
The delivery of the code being done in a regular basis, all the source code had to be built and then tested. Detecting and correcting anomalies in the event of failed builds and testing was difficult and time-consuming, which permanently delayed the software delivery process.	Each commit had to be built and tested. Jenkins allowed developers to focus only on the commit that caused the build to fail instead of testing all the code. The consequence of this is that the developers can deliver new versions of the software more frequently.
By running the tests on all the code, too much time was lost. Developers had to wait for this process to finish before moving forward.	Jenkins provides developers with the test results of each phase running
The developers had to do all this testing process manually	The developers have to commit their changes and Jenkins will do the rest.

Codeception: As for codeception Tests: it combines all testing levels (acceptance, functional, unit), perfect for REST and SOAP API testing and tests can be written in BDD format with Gherkin. Codeception supports all testing types. By default, some tools allow you to set up unit, functional and acceptance tests in a unified framework. However, Codeception does not allow continuous integration, it must be supported by another tool to get it. In addition, its execution is unstable: rendering and JavaScript issues can lead to unpredictable results. Functional tests, JavaScript and AJAX can not be tested and that's why we chose Jenkins on Codeception to be paired with Selenium.

Table 3: Codeception: pros and cons of each testing phase

	Pros	**cons**
Unit Test	Very fast and allows to cover rarely used features Can test the stability of the application kernel essential for all developer	can't test interactions between modules unstable in support: very sensitive to code changes
Acceptance Test	can be run on any website Good support of AJAX	worst performances: requires us to run the browser requests and JavaScript code written can be understandable by clients and managers and that we repopulate database running few tests can lead to many false-positive results Poor execution stability: rendering and JavaScript issues can lead to unpredictable results.
Functional Test	like acceptance tests, but much faster reports provided are very detailed code written can be understandable by managers and clients	Poor support of AJAX requests and JavaScript By simulating the browser, it can lead to more misleading results Requires a framework.

Selenium: Selenium is one of the most popular automated testing tools. Its strength lies in the fact that it can test the functional aspects of web applications and supports a wide range of browsers and platforms.

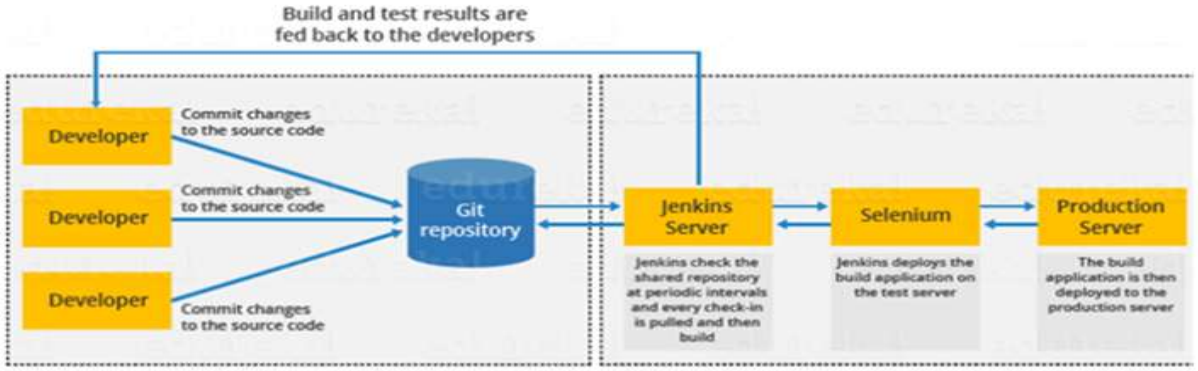
The Selenium suite is composed of four basic components; Selenium Grid, Selenium IDE, WebDriver, Selenium RC. Selenium IDE is Firefox extension that can record and play back web applications tests. Selenium WebDriver communicates directly with the web browser and uses its native ability to automate testing.

Proposed Solution

No tool of the mentioned tools can manage the whole process of testing on its own. Therefore, to overcome such a chaos there was a drastic need for a system to exist where developers can continuously trigger a build and test for every change made in the source code. Jenkins is an open source technology, so the code is open to review and has no licensing costs. This topology ensures a scalable, responsive, and stable environment. Jenkins pushes up code quality by automatically testing within a short period after code commit, and then shouting loudly if build failure occurs. Jenkins is the most mature CI tool available and in the following example, we are going to illustrate how Continuous Integration with Jenkins overcame the above shortcomings along with the use of Selenium framework to guarantee perfection needed in the entire workflow.

First, when a developer commits the code to the source code repository the Jenkins server checks the repository at regular intervals for changes. Soon after a commit occurs, the Jenkins server detects the changes that have occurred in the source code repository. Jenkins will pull those changes and will start preparing a new build, If the build fails, then the concerned team will be notified else If built is successful, then Jenkins deploys the built in the test server (Selenium) to run tests and generate reports properly.

After testing phase, Jenkins generates a feedback and then notifies the developers about the build and test results. It will continue to check the source code repository for changes made in the source code and the whole process keeps on repeating.



4. Conclusion

Software testing tools can be selected based on Application needed to be tested, budget, usage and the efficiency required. Our comparative study is helping to select the suitable tools based on multiple criterion. Producing quality requires a great attention to details. Jenkins is able to detect many of the details and can trigger upstream alerts when failures occur during code build. It offers continuous integration and delivery and can function as a simple CI server. Selenium is an interesting tool when it comes to functional testing. The next step is to compare practical results from the two software and pro-pose a complete architecture in order to get the highest level of efficiency when it comes to test automation.

References

[1] Y. Kumar, "Comparative Study of Automated Testing Tools : Selenium , SoapUI , HP Unified Functional Testing and Test Complete," vol. 2, no. 9, pp. 42–48, 2015.
[2] A. M. F. V. De Castro, G. A. Macedo, and A. C. Dias-neto, "Extension of Selenium RC Tool to Perform Automated Testing with Databases in Web Applications," pp. 125–131, 2013.
[3] G. J. Myers, T. M. Thomas, and J. Wiley, The Art of Software Testing , Second Edition. 2004.
[4] M. Polo, P. Reales, M. Piattini, and C. Ebert, "Test Automation," pp. 84–89.
[5] Y. Amannejad, V. Garousi, R. Irving, and Z. Sahaf, "A Search-based Approach for Cost-Effective Software Test Automation Decision Support and an Industrial Case Study," 2014.
[6] V. Garousi and M. V Mäntylä, "When and what to automate in software testing ? A multi-vocal literature review," vol. 76, pp. 92–117, 2016.
[7] D. Kumar and K. K. Mishra, "The Impacts of Test Automation on Software ' s Cost , Quality and Time to Market," Procedia - Procedia Comput. Sci., vol. 79, pp. 8–15, 2016.
[8] S. Gojare, R. Joshi, and D. Gaigaware, "Analysis and Design of Selenium WebDriver Automation Testing Framework," Procedia - Procedia Comput. Sci., vol. 50, pp. 341–346, 2015.
[9] S. Rajeevan and B. Sathiyan, "Comparative Study of Automated Testing Tools : Selenium and Quick Test Professional .," vol. 3, no. 7, pp. 7354–7357, 2014.
[10] M. R. Angmo and M. Sharma, "International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS) Selenium Tool : A Web based Automation Testing Framework," pp. 351–355, 2014.
[11] S. Singla, "Selenium Keyword Driven Automation Testing Framework," vol. 4, no. 6, pp. 125–129, 2014.