Improving Software Automation Testing Using Jenkins, and Machine Learning Under Big Data

Ali Stouky(✉), Btissam Jaoujane, Rachid Daoudi, and Habiba Chaoui

Laboratoire Génie Des Système, L'équipe ADSI, ENSA de Kénitra, Kenitra, Morocco
ali.stouky@gmail.com, btissam.jaoujane@gmail.com, rachiddaoudil7@gmail.com, mejhed90@gmail.com

Abstract. Software testing is an essential phase of software development life cycle that ensures 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 easily with the help of automated testing tools. This paper provides a feasibility study for the most commonly used testing tools, we conducted a comparative study of five open source tools to determine their usability and effectiveness. Another point discussed in our paper is the use of machine learning under big data in order to make the system intelligent so that tests lend themselves to automation. We will show how can the combination of all these mentioned technologies can help users to decide which strategy to go for to save both cost and time during testing phases.

Keywords: Big data · Machine learning · Test automation
Software testing tools · Functional testing

1 Introduction

Software testing is the process of evaluating a system to check that it satisfies the specific requirements and identify the differences between expected and actual outcomes. The objective behind software testing process is to identify all the defects existing in a software product [1]. Software testing can be done manually or automatically, manual testing is error prone and a time-consuming process it become a bottleneck in the enterprise field when multiple tests are executed daily [2].

For that firms needs to speed up the testing time, cut costs and reduce data maintenance effort by automating the software testing. Automation Testing is a term that refers to the use of testing tools to cut the need of manual or human intervention, repetitive or redundant tasks, and discover defects manual testing cannot expose.

In software development, developers are very good at delivering code that performs the client's needs and within the agreed timeframe. But that often comes at a cost, the code quality is not in the required standard because tests are not conducted as they have to be, some use cases are missing or sometimes there are software regressions caused

J. J. Jung et al. (Eds.): BDTA 2017, LNICST 248, pp. 87–96, 2018.
https://doi.org/10.1007/978-3-319-98752-1_10

by files overwritten accidentally by another developer when there's a team of developers involved. Testing is a task that developers take lightly, they rely on others to do it for them and notify them in case there is a bug. Every time a version of the software is delivered, the project manager is not really sure about what to expect.

And here where automated testing comes to picture. In this paper, we will also show how Big Data and machine learning can help us achieve better results.

1.1 Need of Automated Testing in Big Data

The technological developments in big data infrastructure, analytics, and services allow firms to transform themselves into data-driven organizations. Moreover, Big Data helps companies to achieve higher performance like faster problem-solving issues that used to take years to be solved are now taking less time, hence saving the company various dollars and man-hours.

Imagine if we have ten teams in our company each consists of ten developers and each developer works on 10 features per day and runs their test scenarios and generate their reports. In that case we will have 100 scenarios per team and 1000 per day for the whole company, these data are therefore large and the use of big data will allow us to save data processing time.

For that companies of all sizes need a strategy for big data and a plan of how to collect, use, and protect it, every firm needs to build capabilities to leverage big data in order to stay competitive.

2 Machine Learning (ML) Under Big Data

ML is a core research area of artificial intelligence, whose theme is to emulate human learning activities [3]. It studies methods of identifying current and acquiring new knowledge and improving qualities to realize self-perfection.

Machine Learning Under Big Data Can Help Overcome the Challenges in How Can the Workflow Be Automated by Adopting a Developing Trend Described as Follows:

Transfer Learning: The ability of knowledge transferring and transforming between different scenarios is called transfer learning [3]. The traditional machine learning algorithms usually just solve isolated problems. Transfer learning is based on storing knowledge gained while solving one problem and applying it to a different but related problem to improve the learning performance.

As we can see in the Fig. 1 above, traditional machine learning techniques try to learn each task from scratch, while transfer learning techniques try to transfer the knowledge from some previous tasks to a target task when the latter has fewer high-quality training data [4].

In our case, developers usually run their tests scenarios from scratch every time a build was done, which can be time and effort consuming of course. Thus, by adopting this trend the system will be intelligent and if a developer runs a test it will transfer the results and the scenarios to other nodes making them able to profit from those results and to learn from them.

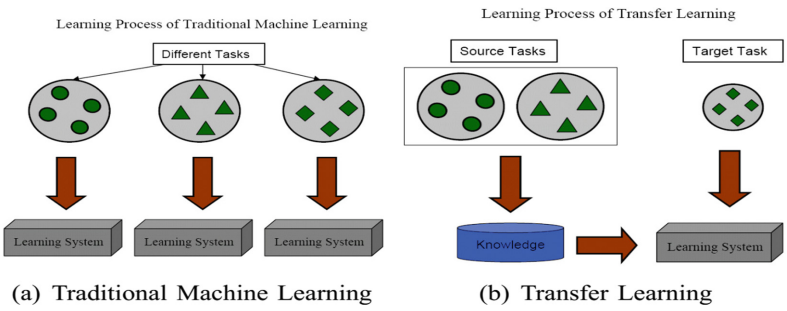

Fig. 1. Difference between Traditional Machine Learning and Transfer Learning

3 Literature Review

Thousands of developers 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. And thus the question that comes to mind is what framework should we use, or how can we decide which solution is better for the company.

In the past 10 years a lot of researches have been done in this scope, Ramler et al. in [5] discussed the benefits extracted from the automated testing. They presented test automation as one solution to reduce testing costs and proposed a cost based model, to decide on automation strategy.

Kasurinen et al. in [6] examined the industrial applications of the automated testing. Theydeduced that approximately 26% of the test cases are automated in industry, the adoption of such approach is a demanding step in software industry and most of the time test automation is used for quality control and quality assurance.

In [8]: Vahid Garousi et al. 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.

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

In [9] Divya Kumar et al. have tried to identify and classify the tools in which the test automation success depends on the three critical software dimensions of time, cost and quality. The statistics and result of their experiments clearly show the positive effects of test automation on cost, quality and time to market of the software.

While in [1], Yogesh Kumar presented a comparative study on testing tools in order to do the selection of right automated software testing tool, they have discussed four

tools that fit into their case study: According to their research Selenium can reduce the cost as it is open source but the efforts involved in scripting for selenium increased by about 15% than other tools. The same for S. Rajeevan that discussed automation testing tools comparing Selenium to Quick Test. As for In [10] Satish have implemented an automation testing framework for testing web applications using selenium WebDriver tool and mentioned that it is useful for dynamically changing web applications.

Another researches [11–13] also have chosen to work with Selenium and that shows that it is commonly used for Functional Testing. To sum up, they mentioned Selenium as it is the most popular open-source tool for Web UI automation testing.

And lastly Sinno Jialin Pan and Qiang Yang in [4] have presented a survey on transfer learning. Beginning from its history, to the relationship between traditional Machine Learning and various transfer learning settings, to its application domains and technologies. They defined what to test how to test it and when in details.

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 our knowledge, has been done to show the difference between the tools who exists nowadays that we can use for continuous integration including all test phases and how can we use them and to reduce the cost what are the tools that are open source and can provide a user based model to run functional testing and managing the use of ML under Big data as we discussed earlier.

4 Evaluation of the Study

Automated testing tools can improve the testing effort if requirements are well defined and managed, and the right tool that matches the needs is selected and most important thing is that tests should lend themselves to automation.

In this section we are going to present five test automation tools which are the most used in the past few years, and in the next section we are going to do a detailed comparison between two of these five tools in order to present the tools that are adequate for our approach.

In our case study we analysed the Integrated Development Environment of two of these tools and performed functional testing of a web application developed with PHP5 using Laravel: the PHP Framework for web artisans and both database management systems: MySQL and MongoDB, also for versions management tool we used Bitbucket a web-based hosting service that is similar to GitHub witch primarily uses GIT and it has three deployment models: Cloud, Bitbucket Server and Data Center.

4.1 Apache Jmeter: Load and Performance Tester

It is an Apache Open source load testing tool, written in Java 6+ and supports all platforms. Recently, Apache released the stable version of Jmeter “v2.11” that supports all platforms. Basically, Jmeter is used for load testing and to analysing and measuring the performance of system/application. It is capable to check the performance of the SOAP, LDAP, Message-oriented middleware (MOM) via JMS,

Mail (SMTP(S), POP3(S) and IMAP(S)), Mongo DB (NoSQL), and Native commands or shell scripts. Its strong GUI design helps in fast building of Test Plan and debugging process.

4.2 Citrus Integration Testing

Automated integration tests for message protocols and data formats! HTTP REST, JMS, TCP/IP, SOAP, FTP, SSH, XML, JSON …Citrus is a testing tool that enables the test team to define whole use case tests to be executed fully automated. Incoming and outgoing messages are predefined in the test case. The tester defines a message flow as it is designed for a use case. All surrounding interface partners are simulated regardless their transport protocols (Http, JMS, TCP/IP, SOAP, and many more).

4.3 Selenium

Selenium suite is comprised of four basic components; Selenium IDE, Selenium RC, WebDriver, Selenium Grid. Selenium IDE is Firefox add-on for record-and-playback web application tests. WebDriver directly communicates with the web browser and uses its native compatibility to automate. It is implemented as a Firefox extension, and allows you to record, edit, and debug tests.

4.4 Codeception

Codeception collects and shares best practices and solutions for testing PHP web applications. With a flexible set of included modules tests are easy to write, easy to use and easy to maintain. Codeception encourages developers and QA engineers to concentrate on testing and not on building test suite. Codeception supports major frameworks: Symfony, Silex, Phalcon, Yii, Zend Framework, Lumen, and Laravel.

4.5 Jenkins

An open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins triggers a build for every change made in the source code repository for example Git repository. Once the code is built it deploys it on the test server for testing. Concerned teams are constantly notified about build and test re-sults. Finally, Jenkins deploys the build application on the production server.

With Jenkins, organizations can accelerate the software development process through automation (Table 1).

Table 1. Comparison of open source automated testing tools

Features	Tools				
	Selenium	JMeter	Citrus	Codeception	Jenkins
Operating System/Platform	ALL	ALL	ALL	ALL	ALL
Browser Support	ALL Browsers	Google Chrome, IE, Mozilla Firefox, Opera	Google Chrome, IE, Mozilla Firefox, Opera	ALL Browsers	ALL Browsers
Language and Frameworks Support	Php, java, C#, javascript, perl, python, Ruby	Java, NodeJS, PHP, ASP .NET	Java, Xml	Php, Ruby, Python, Java, .Net, Appium, NodeJS, Javascript, Robot Framework	Python, Ruby, Java, Android, C/C++
Ease Of Use	Needs a quite Expertise	No coding is necessary at the basic level	Experience needed	Experience needed	Very easy to use
Software Cost	Open Source	Licence Apache	Licence Apache	Open Source	Open Source
Type Tests	Unit, functional	Performance	Automated integration tests	Acceptance, functional, unit	Unit, Automated integration tests
Script Creation Time	Low	High	High	High	Low
Report Generation	Integration with jenkins can give good reporting & dashbord capabilitie	Graphic, spline, assertion result, tree, statistics	test plan and document test coverage	Several information is provided: execution time, statistics	Checkstyle, PHPMD, PHPCPD, HTMLReport…

5 Analysis of Study

Even if it is known by its Ease of use Jmeter doesn't work well when you have frequent components that you reuse across tests or having different modular tests chained together to form a bigger load tests. It gets harder to do so as you progress with more tests or more levels of testing. The lazy way most beginners would deal with reuse is copy & paste from one area to another or one file to another witch can affect the efficiency of the work.

As for Citrus, it is an integration testing platform for testing live applications deployed in a target environment. In our case we aim to do functional testing as long as continuous integration which cannot be done with Citrus. Also based on the benchmark table and what have been said previously we decided to analyse three of the proposed tools:

5.1 Codeception

First of all, Codeception combine all testing levels. Out of the box you have tools for writing unit, functional, and acceptance tests in a unified framework. Perfect for REST and SOAP API testing and tests can be written in BDD format with Gherkin also it allows multi-request functional tests. In the Table 2 below we present in details its pros and cons for each testing phase.

Table 2. Codeception: pros and cons of functional testing phase

	Pros	Cons
Functional test	• Can be run on any website • Can provide more detailed reports • You can still show this code to managers and clients • Stable enough: only major code changes, or moving to other framework, can break them	• JavaScript and AJAX can't be tested • By emulating the browser you might get more false-positive results • Requires a Framework • Fewer checks can lead to false positive results

5.2 Jenkins

Jenkins can automate the building of software regularly, and trigger tests pulling in the results and failing based on defined criteria. Failing early through build failure lowers the costs, increases confidence in the software produced, and has the potential to morph subjective processes into an aggressive metrics-based process that the de-velopment team feels is unbiased (Table 3).

Table 3. Comparison between "Before and After Jenkins"

Before Jenkins	After Jenkins
The entire source code was built and then tested. Locating and fixing bugs in the event of build and test failure was difficult and time consuming, which in turn slows the software delivery process	Every commit made in the source code is built and tested. So, instead of checking the entire source code developers only need to focus on a particular commit. This leads to frequent new software releases
Developers have to wait for test results	Developers know the test result of every commit made in the source code on the run
The whole process is manual	You only need to commit changes to the source code and Jenkins will automate the rest of the process for you

5.3 Selenium

Because of its many advantages, Selenium finds wide usage for UI, regression, unit and acceptance testing. But even selenium is the most popular tool it stills not a complete, comprehensive solution to fully automating the testing of web applications. It requires third-party frameworks, language bindings and another configuration to be truly effective.

6 Proposed Solution

It is obvious from the above stated cons and issues that not only the software delivery process became slow but the quality of software also went down. This leads to customer dissatisfaction and no tool will manage the whole process of Testing along with continuous integration and continuous deployment. So 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.

On top of Jenkins is an open source technology, so the code is open to review and has no licensing costs. And it is a master slave topology that distributes the build and testing effort over slave servers with the results automatically accumulated on the master. This topology ensures a scalable, responsive, and stable environment.

Jenkins is the most mature CI tool available and in the following diagram 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 (Fig. 2).

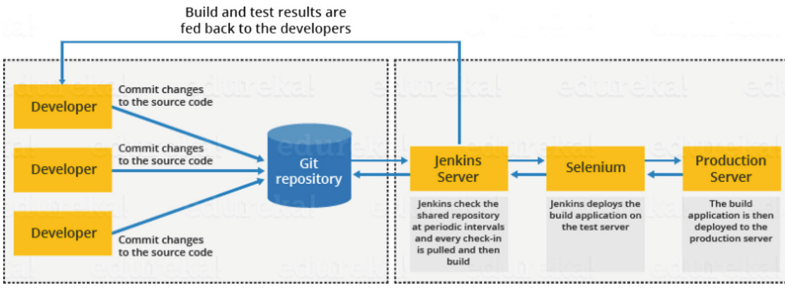

Fig. 2. Generic flow diagram of Continuous Integration with Jenkins

- 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.

The other important key point as we discussed earlier is the combination of this approach of testing with the use of Big data to have a solid architecture so that machine learning can have a basic source of information or outcomes to use them as incomes, and hence to learn from experiences, findings and mistakes of those tests who have previously been passed.

The Learning process from those results transforms the system to become intelligent so the testing phases work fluently and in a predictive way too. Machine Learning needs to develop and progress to change big data into actionable insight. On one side, big data provides rich information for Machine Learning algorithms to pick up underlying patterns and to build predictive models; and on the other, traditional Machine Learning algorithms face crucial challenges like scalability to release the true value of big data.

7 Conclusion

Software testing tool 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. It presents each tool with features that are in the same and different degree with other tools mentioned and how each one behaves against others tools' characteristics.

Producing quality requires a great attention to details. Jenkins can pay attention to many of the details and shout loudly when violations occur.

In this paper, we have also shown how the adoption of big data can help a company to achieve better results and high performance in order to reduce time and cost and how firms nowadays must go for big data technologies, moreover optimizing tests execution time and effort by using Machine learning algorithms to automate the whole process and to make the system intelligent. Our future work will encounter more tools and more technologies also that will help in building a user based smart-model.

References

1. Kumar, Y.: Comparative study of automated testing tools : selenium, SoapUI, HP unified functional testing and test complete. **2**(9), 42–48 (2015)
2. De Castro, A.M.F.V., Macedo, G.A., Dias-neto, A.C.: Extension of Selenium RC Tool to Perform Automated Testing with Databases in Web Applications, pp. 125–131 (2013)
3. Shi, C., Wu, C., Han, X., Xie, Y., Li, Z.: Machine learning under Big Data, no. Emim, pp. 301–305 (2016)
4. Pan, S.J., Fellow, Q.Y.: A survey on transfer learning, pp. 1–15 (2009)
5. Ramler, R., Wolfmaier, K.: Economic perspectives in test automation: balancing automated and manual testing with opportunity cost, pp. 85–91 (2006)

6. Kasurinen, J., Taipale, O., Smolander, K.: Software test automation in practice : empirical observations, vol. 2010 (2010)
7. Amannejad, Y., Garousi, V., Irving, R., Sahaf, Z.: A Search-based Approach for Cost-Effective Software Test Automation Decision Support and an Industrial Case Study (2014)
8. Garousi, V., Mäntylä, M.V.: When and what to automate in software testing ? A multi-vocal literature review, vol. **76**, 92–117 (2016)
9. Kumar, D., Mishra, K.K.: The Impacts of test automation on software' s cost, quality and time to market. Procedia Comput. Sci. **79**, 8–15 (2016)
10. Gojare, S., Joshi, R., Gaigaware, D.: Analysis and design of selenium webdriver automation testing framework. Procedia Comput. Sci. **50**, 341–346 (2015)
11. Rajeevan, S., Sathiyan, B.: Comparative study of automated testing tools : selenium and quick test professional, **3**(7), 7354–7357 (2014)
12. Angmo, M.R., Sharma, M.: International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS) Selenium Tool : a web based automation testing framework, pp. 351–355 (2014)
13. Singla, S.: Selenium keyword driven automation testing framework, **4**(6), 125–129 (2014)