JCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-6, Issue-7, July 2018 E-ISSN: 2347-2693

Comparative Study of Selenium WebDriver and Selenium IDE (Integrated Development Environment)

Shilpa Garg[1*], Paramjeet Singh[2], Shaveta Rani[3]

[1] Dept. of CSE, Giani Zail Singh Campus College of Engineering and Technology, Bathinda, India
[2] Dept. of CSE, Giani Zail Singh Campus College of Engineering and Technology, Bathinda, India
[3] Dept. of CSE, Giani Zail Singh Campus College of Engineering and Technology, Bathinda, India

[*]*Corresponding Author: shilpagarg14.9@gmail.com, Tel.: +91-99144-00998*

Available online at: www.ijcseonline.org

Accepted: 19/July/2018, Published: 31/July/2018

Abstract— Testing is necessary because we all make mistakes. Additionally, we are more likely to make errors when dealing with perplexing technical or business problems, complex business processes, code or infrastructure, changing technologies, or many system interactions. This is because our brains can only deal with a reasonable amount of complexity or change when asked to deal with more our brains may not process the information we have correctly. Some of these errors are not important, but some of them can be expensive and damaging, with loss of money, time or corporate reputation and may even cause injury or death. A key element to conduct successful software testing, are various testing tools. In addition to tool support for regressive testing, selection of appropriate tool also becomes equally important depending upon the cost involved in terms of skillset required and maintenance of test scripts.

Keywords—Test Automation, Selenium IDE, Selenium Webdriver, Mutation Rate, Error Rate

I. Introduction

Software testing ensures quality and reliability of the software do not fall under risk levels that organization has benchmarked. Test execution can be performed in two ways:

Manual: Tester ensures application's correct behavior where he/she behaves as the end user and performs test steps manually.

Test Automation: Tasks like comparing contents of large data file or simulating how system would behave might overload a person and even prone to mistakes as they soon get bored. In test automation, software tools take up the burden of performing repetitive tasks making test execution more efficient and more reliable and even faster than doing it manually.

Rest of the paper is organized as follows, Section I contain introduction to software testing and its types, Section II contain related work of selection and importance of test execution tools, Section III explain Selenium and its components, Section IV explain the methodology of creation and comparison of test scripts in Selenium WebDiver and Selenium IDE, Section V describe results based on test scripts generated for dummy web applications and Section VI concludes research work with future directions.

II. Related Work

Customers have indicated that testing tool selection is challenging process and look up for recommendations from testing consultants who daily face various testing tools at customer premises. Upon starting a new project testing consultants must go through existing tools and quickly learn new tools as well to pick and recommend most appropriate one.

Selecting most appropriate test execution tool helps in improved staff performance and time savings, improved test results accuracy and earlier bugs identifications before The users are impacted.

A test execution tool most often runs tests that have already been run before. Whenever an existing system is changed (e.g. for a defect fix or an enhancement), all the tests that were run earlier could potentially be run again, to make sure that the changes have not disturbed the existing system by introducing or revealing a defect.

III. SELENIUM

Several commercial and open source tools are available and selenium is possibly the most widely-used open source solution for the formation of test scripts that are executed against the web application under test.

Selenium is composed of multiple software tools that are: -

A. Selenium 2 (aka. Selenium WebDriver): Selenium-WebDriver has been developed to better support dynamic web pages in which the elements of a page can change without reloading the page. WebDriver's goal is to provide a well-designed, object-oriented API that provides advanced support for today's advanced Web application test problems. It is compatible with many browsers such as Firefox, Chrome, IE and Safari. Selenium RC + WebDriver = Selenium 2.0

B. Selenium 1 (aka. Selenium RC or Remote Control): Selenium RC was the main project of Selenium for a long time, before the fusion WebDriver / Selenium brought Selenium 2, the newest and most powerful tool.

C. Selenium IDE: Selenium IDE (Integrated Development Environment) is an easy-to-use Chrome and Firefox extension and is generally the most efficient way to develop test cases. It records the user's actions in the browser for you, using existing Selenium commands, with parameters defined by the context of that element. This is not only a time-saver, but also an excellent way of learning Selenium script syntax. Specifically, Selenium IDE does not provide iteration or conditional statements for test scripts.

D. Selenium-Grid: Selenium Grid allows you to run tests in parallel, i.e. different tests can be performed simultaneously on different remote machines. This has two advantages. First of all, if you have a large test set or a slow-running test set, you can significantly increase performance by using Selenium Grid to split the test set and run different tests at the same time using the different machines. In addition, if you need to run the test suite in multiple environments, you can have several remote computers that support and run the tests at the same time. In any case, Selenium Grid greatly improves the time needed to run the suite using parallel processing.

IV. METHODOLOGY

To deliver results accurately, test data should be organized wisely with selection of appropriate automation framework followed by the test script creation in both Selenium IDE and Selenium WebDriver. After the formation of test scripts, impacted areas in the test suite will be analyzed based on changes made.

A. Comparison between Selenium WebDriver and Selenium IDE (Integrated Development Environment)

Most test execution tools offer a way to start capturing or recording manual tests; therefore, they are also known as "capture / reproduction" tools, "capture / reproduction" or "recording / reproduction" tools. The analogy is to record a television program and play it. However, tests are not something that is only reproduced to allow someone to see the tests interact with the system, which can react slightly differently when the tests are repeated. Therefore, the acquired tests are not suitable if you want to achieve long-term success with a test execution tool. The test execution tools use a scripting language to guide the tool. The scripting language is a programming language. Therefore, any tester who wishes to directly use a test execution tool must use programming skills to create and edit scripts. The advantage of programmable scripts is that tests can repeat actions (cycles) for different data values (e.g. Test entries), they can follow different paths depending on the result of a test (for example, if a test fails, go to a different set of tests) and can be called by other scripts giving a certain structure to the test set. The captured script is very difficult to maintain because:
1. It is closely related to the flow and interface presented by the GUI.
2. You can trust the circumstances, status and context of the system at the time the script was recorded. For example, a script will acquire a new order number assigned by the system when a test is recorded. When that test is performed, the system will assign a different order number and will reject subsequent requests that contain the previously acquired order number.
The test input information is "coded", that is, it is incorporated into the single script for each test. Any of these things can be overcome by editing the scripts, but we're not just recording and playing! If more time is needed to update an acquired test than is necessary to run the same test again manually, the scripts tend to be abandoned and the tool becomes "shelf-ware".

B. Computation Parameters

1) Mutation Rate: Gives the rate at which mutations need to be made in the entire test suite depending upon the change request in either data field, locator or verification point.

$$\text{Mutation Rate} = \frac{\text{No of Changes}}{\text{No of Iterations}} * 100$$

2) Error Rate: Gives the percentage of wrong entries made while manually editing test scripts (0.01 value is opted based on Six Sigma techniques)

$$\text{Error Rate} = \frac{0.01 * \text{No of Changes}}{\text{No of Iterations}} * 100$$

For Selenium IDE, No of Changes = No of Corrections * No of Iterations
For Selenium WebDriver, No of Changes = No of Corrections

3) Accuracy: Gives the efficiency and correctness of the test suite after making changes to the test scripts.

$$\mathcal{A}ccuracy = 100\% - \mathcal{E}rror\ \mathcal{R}ate$$

More the accuracy, more reliable is the test suite in terms defects detection. Accuracy can be improved by reducing the chances of errors while altering the scripts manually which is directly proportional to total changes required to be made in the test scripts. Judicious selection of automation tool is critical to ensure the success of the testing project that can be based on above listed parameters.

V. RESULTS AND DISCUSSION

Below are the results documented for five web application:

A. Web Application 1: Test scripts are formed and analysed in Selenium IDE and Selenium WebDriver based on the following data.

No of Test Data Iterations: 50
No of Locators: 10
No of corrections made to Locators: 2

Table 1. Analyzation results for web application 1

Parameters	Selenium WebDriver	Selenium IDE
Mutation Rate	(2/50) * 100 = 4%	[(50 * 2)/50] * 100 = 200%
Error Rate	[(0.01 * 2) / 50] * 100 = 0.04%	[(0.01 * 2 * 50) / 50] * 100 = 2%
Accuracy	100% - 0.04% = 99.96%	100% - 2% = 98%

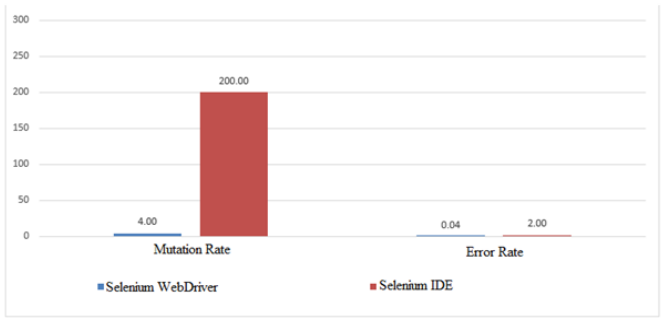

Figure 1. Analyzation results for web application 1

B. Web Application 2: Test scripts are formed and analyzed in Selenium IDE and Selenium WebDriver based on the following data.

No of Test Data Iterations: 60
No of Verification Points: 20
No of corrections made to Verification Points: 3

Table 2. Analyzation results for web application 2

Parameters	Selenium WebDriver	Selenium IDE
Mutation Rate	(3/60) * 100 = 5%	[(60 * 3)/60] * 100 = 300%
Error Rate	[(0.01 * 3) / 60] * 100 = 0.05%	[(0.01 * 3 * 60) / 60] * 100 = 3%
Accuracy	100% - 0.05% = 99.95%	100% - 3% = 97%

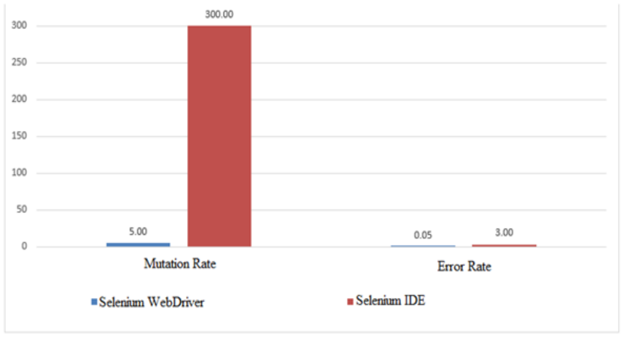

Figure 2. Analyzation results for web application 2

C. Web Application 3: Test scripts are formed and analysed in Selenium IDE and Selenium WebDriver based on the following data.

No of Test Data Iterations: 70
No of Data Fields: 30
No of corrections made to Data Fields: 7

Table 3. Analyzation results for web application 3

Parameters	Selenium WebDriver	Selenium IDE
Mutation Rate	(7/70) * 100 = 10%	[(70 * 7)/70] * 100 = 700%
Error Rate	[(0.01 * 7) / 70] * 100 = 0.1%	[(0.01 * 7 * 70) / 70] * 100 = 7%
Accuracy	100% - 0.1% = 99.9%	100% - 7% = 93%

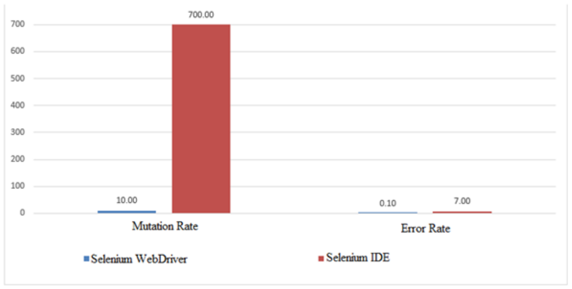

Figure 3. Analyzation results for web application 3

D. Web Application 4: Test scripts are formed and analysed in Selenium IDE and Selenium WebDriver based on the following data.

No of Test Data Iterations: 80
No of Verification Points: 40
No of corrections made to Verification Points: 4

Table 4. Analyzation results for web application 4

Parameters	Selenium WebDriver	Selenium IDE
Mutation Rate	(4/80) * 100 = 5%	[(80 * 4)/80] * 100 = 400%
Error Rate	[(0.01 * 4) / 80] * 100 = 0.05%	[(0.01 * 4 * 80) / 80] * 100 = 4%
Accuracy	100% - 0.05% = 99.95%	100% - 4% = 96%

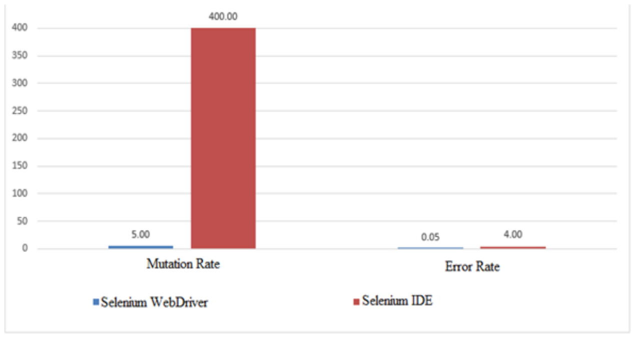

Figure 4. Analyzation results for web application 4

E. Web Application 5: Test scripts are formed and analysed in Selenium IDE and Selenium WebDriver based on the following data.

No of Test Data Iterations: 90
No of Locators: 50
No of corrections made to Locators: 6

Table 5. Analyzation results for web application 5

Parameters	Selenium WebDriver	Selenium IDE
Mutation Rate	(6/90) * 100 = 6.67%	[(90 * 6)/90] * 100 = 600%
Error Rate	[(0.01 * 6) / 90] * 100 = 0.067%	[(0.01 * 6 * 90) / 90] * 100 = 6%
Accuracy	100% - 0.067% = 99.933%	100% - 6% = 94%

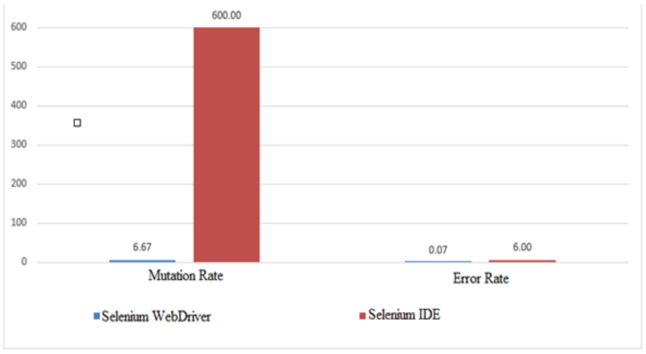

Figure 5. Analyzation results for web application

VI. CONCLUSION and Future Scope

For a test execution tool to offer benefits, it must meet a need within the organization and resolve this need effectively and efficiently. Furthermore, the test execution tool should help to exploit the strengths of the organization and address its weaknesses. The organization must be prepared for the changes that will be provided with the new test execution tool. If current testing practices are not good and the organization is not mature, it is generally more convenient to improve testing practices than to try to find tools that support bad practices.

Selenium IDE and Selenium WebDriver test scripts for dummy web applications are compared based on user-friendliness, technical skills required, re-usability of scripts, alteration of scripts, handling of multiple test data, duplicity of scripts, component based approach and maintenance of scripts.

Table 6. Comparing Selenium WebDriver and Selenium IDE (Integrated Development Environment)

Parameter	Selenium IDE	Selenium WebDriver
Scripts Alteration	High	Low
Scripts Duplicity	High	Low
Test Suite User Friendliness	High	Low
Scripts Re- Usability	High	Low
Handling Multiple Test Data	High	NA
Scripts Maintenance	High	Low
Technical Skills Required	Low	High

REFERENCES

[1] R. Chauhan, I. Singh, "*Latest Research and Development on Software Testing Techniques and Tools*", INPRESSCO International Journal of Current Engineering and Technology, 2014.
[2] S. P, D. N, "*Automation of Software Testing in Agile Development - An Approach and Challenges with Distributed Database Systems*",

GJRA - GLOBAL JOURNAL FOR RESEARCH ANALYSIS, Vol. 3, pp.1-7, 2014.

[3] N. Bhateja, "*A Study on Various Software Automation Testing Tools*", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 5, pp.1-6, 2015.

[4]. S. Sharma, "*Study And Analysis Of Automation Testing Techniques*", Journal of Global Research in Computer Science, Vol. 3, pp.1-12, 2012.

[5] "*Analysis of Automation and Manual Testing Using Software Testing Tool*", IJIACS, Vol. 4, 2017, ISSN ISSN 2347 − 8616.

[6] S. Thummalapenta, S. Sinha, N. Singhania, "*Automating Test Automation"*, IBM T. J. Watson Research Center IBM Research-India, Vol. 2017.

[7] "*An Approach of Software Design Testing Based on UML Diagrams*", International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, pp.1-2, 2014.

[8] "*A Unique Technique to Handle the Complexity and Improve the Effectiveness of Test Cases in Software Testing*", Journal of Innovative Technology and Education, Vol. 2, 2015.

[9] M. Dande and N. Galla, "*Automation Testing Frameworks for SharePoint application*", International Journal of Computer Sciences and Engineering, Vol.3, Issue.11, pp.33-38, 2015.

[10] R. Sharma, R. Dadhich, "*Implications of Software Testing Strategies at Initial Level of CMMI: An Analysis*", International Journal of Computer Sciences and Engineering, Vol.6, Issue.5, pp.1055-1061, 2018.

[11] N. Kaur, J. Kaur, J. S. Budwal, "*Application of ACO in Model Based Software Testing: A Review*", International Journal of Computer Sciences and Engineering, Vol.6, Issue.3, pp.370-374, 2018.

[12] B. Saha, D. Mukherjee, "*Analysis of Applications of Object Orientation to Software Engineering, Data Warehousing and [13] Teaching Methodologies*", International Journal of Computer Sciences and Engineering, Vol.5, Issue.9, pp.244-248, 2017.

[14] N. Sudheer, V. Sarma, N. Ahmad, "*Implementing Different Types and Variants for Software Testing Process and Techniques*", International Journal of Computer Sciences and Engineering, Vol.5, Issue.4, pp.34-39, 2017.

[15] A. Verma, A. Khatana, S. Chaudhary, "*A Comparative Study of Black Box Testing and White Box Testing*", International Journal of Computer Sciences and Engineering, Vol.5, Issue.12, pp.301-304, 2017.

[16] N. Sudheer, S.H. Raju, "*Different approach Analysis for Static Code in Software Development*", International Journal of Computer Sciences and Engineering, Vol.4, Issue.9, pp.111-118, 2016.

[17] R. K. Sahoo, D. P. Mohapatra, M. R. Patra, "*A Firefly Algorithm Based Approach for Automated Generation and Optimization of Test Cases*", International Journal of Computer Sciences and Engineering, Vol.4, Issue.8, pp.54-58, 2016.

[18] N. Sudheer, V. Sharma and S. H. Raju, "*A Process Web Application Testing Using TAO Tool Search Based Genetic Algorithm*", International Journal of Computer Sciences and Engineering, Vol.4, Issue.7, pp.94-100, 2016.

[19] S. Kannan, T. Pushparaj, "*A study on variations of Bottlenecks in Software Testing*", International Journal of Computer Sciences and Engineering, Vol.2, Issue.5, pp.8-14, 2014.

[20] S. Bharti and S. N. Singh, "*Improvised Agile SCRUM Using Test-Asa-Service*", International Journal of Computer Sciences and Engineering, Vol.3, Issue.3, pp.166-171, 2015.

Authors Profile

Ms. S. Garg pursed Bachelor of Information Technology from Rayat Bahra University, Mohali, India in 2013 and currently pursuing Master of Computer Science from Giani Zail Singh Campus College of Engineering and Technology, Bathinda, India. Her main research work focuses on Quality Assurance, Software Engineering, Automation Testing, SDLC, Static and Dynamic Testing, Agile Methodologies, Test Management.