COMPARISON BETWEEN COMBINED AND DATA DRIVEN FRAMEWORK FOR WEB APPLICATIONS IN WEBDRIVER

[1]Oshin, [2]Ashima Arya
[1]M.TECH, [2]Assistant Professor
[1]Department of Computer Science & Engineering,
[1]Deenbandhu Chhotu Ram University of Science and Technology (Murthal), Sonepat, INDIA

Abstract: Combined Automation Framework (CAF) and Data Driven Framework (DDF) are typical frameworks for Web Applications in WebDriver. This paper presents a comparison of the CAF with a DDF in Selenium WebDriver test automation based on various parameters. It is evident from the simulation results that the CAF framework works better regarding terms of reusability, script efficiency, client interaction, compatibility with other scripting languages and its capability to get ported to other platforms. DDF performs well in case of large Test Data repository. It is more successful when used in Data Centric testing**.**

***Index Terms* - DDF, KDF, CAF, HAF**

I. INTRODUCTION

Data-Driven scripts are the application specific scripts written in the tester's suitable language. They can be modified as per the varying data input. User specific names are used for key application input fields and selection of programs allowing the script to drive the application with external data supplied by the calling function that invokes the test script. Keyword-Driven Framework (KDF) is based on high level of abstraction of the individual test cases, which means it focuses on specifications from the client. It is all about the set of keywords that can enable the business/non-technical persons to write test cases without even knowing the details of implementation of Application Under Test (AUT) [2]. The other benefits that come from applying high-level tests mainly on systems or acceptance tests are its focus on business and systems requirement without concentrating on concrete implementation approaches. The advantage of automation support of test execution allows execution of regression testing, i.e., script run to check that changes that is introduced during systems maintenance and evolution. This allows the tests to be easily adjusted to the changing requirements. One can easily customize Hybrid Automation framework into a framework that works basically on Data-Driven Framework (DDF) and Keyword-Driven Frameworks (KDF). The Combined Automation Framework (CAF) combines the advantages of both the DDF and KDF and are was discussed in [13] [14]. This framework tries to match up with Hybrid Automation Framework (HAF). Further, the comparison of CAF with the DDF can highlight the pros and cons of both these frameworks. Thinking of CAF far superior to the DDF in every aspect is not very concise. Both have their own set of advantages.

The contribution of the paper is as under:

- The paper details state-of-the-art automation frameworks for Web Applications in WebDriver.
- Further, the paper presents a comparison of the CAF with a DDF in Selenium WebDriver test automation based on various parameters such as reusability, script efficiency, client interaction, compatibility, Test script maintenance, Test robustness and performance z[12].

The rest of the paper is organized as under:

Section II presents details of the state-of-the-art automation frameworks for web applications in webdriver. The CAF & DDF detailed in Section III. The comparison between CAF & DDF based on various parameters is presented in section IV. The paper is concluded in section V. In the last, the references are listed.

II. STATE-OF-THE-ART AUTOMATION FRAMEWORKS

The evolution of the Automation tools from framework-less scripting and Record & Run feature to developing them into complete execution suites for the ease of automation shows the awareness of the developers [1] [9]. This paper compares CAF which is based on the idea of keywords and data inputs with solely the DDF. Although both the frameworks create test scripts on a common platform and there is no need of any programming knowledge while executing on them. Testers needed to alter the scripts on the common platform and run them to automatically test the application. Tests were developed with data table inputs and outputs along with specified action-words to execute the scripts in CAF. The added task that was done in DDF was creating large test data for each individual test case or test script. The objective was to compare the CAF (having keyword with meaningful name and functions associated with these keywords are common to all applications) which would be independent of Application with the DDF (where the use of action words was not encouraged, only data was input from various types of files). The end goal was to check the parameters in which CAF was better than DDF and vice versa. These frameworks have been executed with Selenium WebDriver - which is open source tool. A driving-script was used in both CAF and Data-Driven framework technique to control the execution from one Test itself. CAF imported the keywords and data located in Excel/CSV/flat file and the reporting was done using Junit [8].

2.1. Conventional Framework:

Each application may seem unique but its underlying components are same. This brings forth the need for developing a framework which focuses on the baseline components and not the conventional application. Following this will get us rid of becoming application specific and we can reuse almost everything we develop for every application through the automated test process. Almost every application comes with some form of menu system [4].

There are buttons to push, boxes to check, lists to view etc. For a typical automation tool script there is a very small number of functions for each type of component. These functions work independent of their containing applications with the component objects[5].

Conventional automation scripts somewhere call these component methods. Therefore, we already have tools to achieve this application independence. But our scripts are using the application specific routine calls and hard coded values thus reducing their effectiveness as application-independent constructs. Further, the methods are susceptible to failure unless a very specific application state or synchronization exists at the time they are executed [2]. There is no error correction built with these methods. To deal with such failure in traditional scripts we must place extra code before or after a set of commands, to ensure the proper application state and synchronization is maintained with small awareness's like the window has the current focus, the component that we want to select, or press, or edit exists and is in the proper state. After which one can perform the required operation and separately verify the result of our actions. To achieve maximum robustness, we would have to code these tests (of their synchronization and state) for each component function call in our scripts. But the stress is, we can never complete this task in real life scenario. It will make the scripts lengthy, difficult to maintain and unreadable. This increases the possibility that script will fail. Here lies the need for application-independent framework [2]. This will reduce our efforts every time we write a new script, and execute it for every call to any component function. The added functions this framework must handle will be- verifying the element of interest is in the proper state, doing something with that element, details of insuring the correct window, and logging the success or failure reports. All that our framework will need are the variables and application-specific data to our application-independent framework. Therefore, we will provide our completed test designs as executable input into our automation framework. However, if we can limit the percentage of our application specific test scripts, while taking advantage of the best features of our automation framework, we will reap the rewards in every project arena.

2.2. Data Driven Framework (DDF):

In Data-Driven methodology, large amount of test data repository is maintained and is separated from test scripts. Results are then returned against the test data using validations or checkpoints. Entire test data is created from some external files like excel, Flat file, CSV or XML or even any other database table. If all the test data combinations are pass, then only the test case is treated as "PASS". Entire test can be considered "FAIL" due to one test data combination failing (Figure 2.1). It is suitable for test cases with multiple test data combinations since it reduces the number of test scripts needed to implement all the test cases. Before any test execution it is imperative to have the test data prepared. This framework is not recommended for the very simple actions which do not have any test data combination to be used. A Data-Driven framework can sometimes use the action words and import the test data from external files. This feature allows it to function like KDF (Figure 2.2).

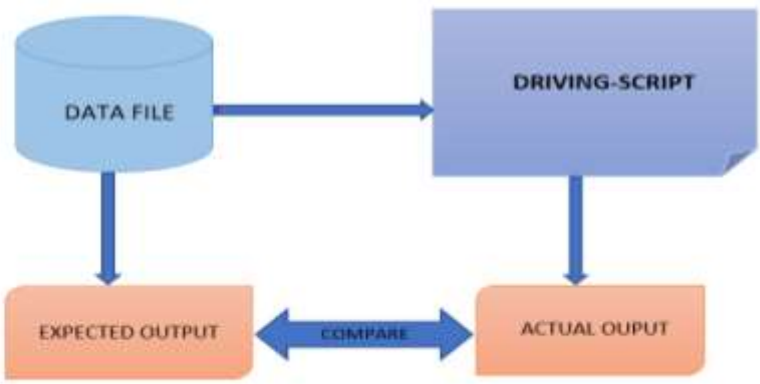

Figure 2.1: Model of Data-Driven Framework

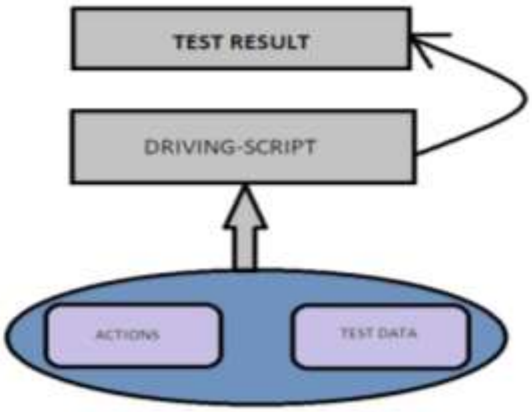

Figure 2.2: Model of Data-Driven Framework with Action words

2.3. CAF Test Framework:

It is a methodology using the advantage of both Data, Modularity and Keyword driven framework. The CAF framework is built with a number of reusable modules and function libraries that are developed keeping in mind the features like management for effective test design, execution, and traceability; Maintenance – Known to reduce maintenance effort; Reusability – It allows to reuse test cases and library functions; Access to test – easy to design, develop, modify and debug test cases while executing; Test scheduling – Allows to schedule automation execution; Reporting & Customization – customizable reporting of test results ensure the quality output; Error handling – due to advanced error handling and scenario recovery; Flexibility – framework independent of system or environment under test. The Data-Driven framework is a part of the CAF module (as in Figure 3.1). This implementation of D-D framework is a sub-part of the same process. However, its full implementation is somewhat a separate process, which entirely focusses on different data combinations. On the other hand, the framework can use scripts to perform some tasks that might be too difficult to re-implement in a pure keyword driven approach.

III. CAF VERSUS DDF

CAF consists of any or all of the following sub-components: Libraries, Object Repository, POM (Page Object Model) Data-Driven Technique, Function Parameters, Constant Variables, Action-Driven Technique, Defect reporting mechanism, JUnit Reporting, User specified methods, Error and Exception Handling.

Its focus is to use the advantages of both Data-Driven and Keyword Driven frameworks and therefore named this framework as Combined (Keyword + Data) framework. We are using CAF to extract features of Hybrid framework with less concentration on managing every detail in a test suite and still developing one which is compact in itself. It is homemade and readymade to reduce the drawbacks of solely Data-Driven framework like need of familiarity with programming language, timely emphasis learning curve, more time consumption in terms of data fetch and manipulation by the suite. Additionally, it works to remove the individual drawbacks of Keyword Driven framework also like lesser data driven support, lesser flexibility, high dependency on the developed framework as to where and how the call methods are used, higher time consumption due to communication with the functions time and again and lesser change support due to data interference.

The very basic set of points is to be considered first. The combined framework should be designed in such a way that it is reusable, easily scalable, maintainable, easy to modify from developer's point of view and easy to interact with from user's point of view. The code should be easy to understand and maintain. Grouping of the components must be done separately. Thus, there has to be a basic folder structure to keep the test cases, test data sheets, libraries and test reports grouped together in an effective and efficient manner. The result presentation reports are – summarize report and detailed report. The summary report is Data Excel sheet with the test case names and alongside their execution status. In addition, we can customize this further by providing the link path of the report generated for each test case. There can always be more parameters added to should be able to configure without having to touch the code. In addition, with just a single click of a button, the report gets generated. If required there should be certain functionalities in your framework that it should execute all the test cases. In most of the cases, this feature is implemented using MS Excel Macros. These features indicate the compact capacity of a Framework. All the test validations become just a click away with reduced code interaction. If the framework becomes customer/user friendly, this means that it has achieved more than half of its efficiency. Then the reusability, application independence features come handy.

CAF has been developed using Selenium-WebDriver, a widely accepted web application automation tool and is not much different from Data-Driven framework. It shrinks test cycle times and related costs. Selenium is a platform independent software testing framework for web applications. Selenium can be deployed on Windows, Linux and Mac. The tests can be run directly in most modern web browsers.

CAF increases automation efficiency by minimizing initial coding effort whereas DDF handles the External Data inputs. It is a script-less framework used for test automation of web applications that can be developed on various programming languages like JAVA, .Net, Java, PHP. The framework provides a platform to implement Data-Driven and Keyword-Driven framework by excel template (here, particularly) [11]. It can be used in any current automation project. CAF helps organizations speed up testing at the test design layer while keeping the automation suite flexible to interface with many commercial tools, whenever needed and D-D keeps the data usage flexible. The CAF test framework whereas, provides a comprehensive reporting dashboard for managing tests.

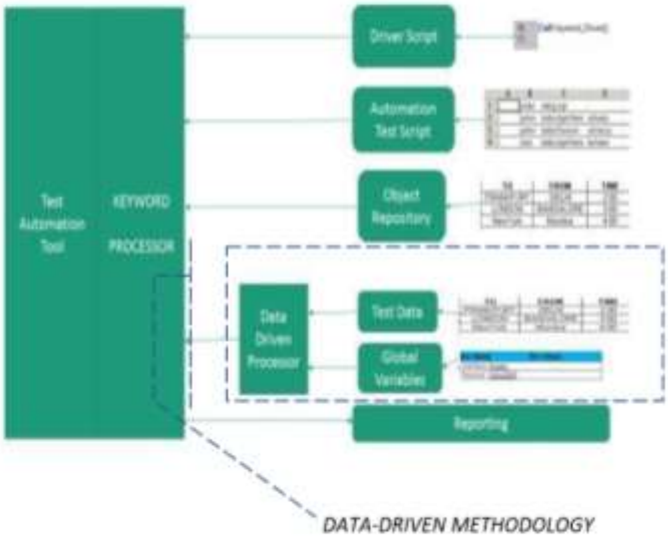

Figure 3.1: Model of Combined Automation Framework

IV. Performance Comparison of CAF with DDF

The main objective of the comparison in this paper is to figure out which framework performs better and in which aspect [1][6]. The modular aspect is important since it checks the GUI feature of the framework. What happens inside one is of no concern to the others. With the black-box approach along with its modularity, the functionality available within each module can be expanded without affecting any other part of the system. This makes code maintenance much simpler. In addition, the complexity of any of the module is minimized. However, modularity alone will not be enough to ensure a highly maintainable framework. The module design must be fully documented and published like any good software project. It will be very difficult for anyone to decipher what it is the framework is designed to do without adequate documentation [3]. The departure of the original framework designers removes the hope of maintenance of the framework. Our test automation efforts will eventually become another negative statistic. To prevent this, proper documentation standards and templates must be defined. Wherever possible, module documentation should be developed "in-context". That is, directly in the source code itself. It will also provide those doing the code maintenance quite a ready reference. Nearly everything they need to know should exist right there in the code [7]. This application usage needs to be equally applied for both Data-Driven and CAF frameworks. To put something in place that people will use and reuse for as long as it is technically efficient. The process begins with establishing connection with the database so that the data driven framework comes into picture and gets applied with the help of the workflow. Next step is the beginning of the Keyword routine so that the Keyword driven framework comes into picture. This will include associating both the Utility and application specific methods with the main script. Next step will be to use the JUnit in order to generate the reports and logs of the test under run. This is basically a plug-in which generates the test run report for the selected test. Then, we need to start the timer variable to keep track of the execution time. Then follows the procedure of setting the Boolean variable for the keywords to be verified with the database and calling the keywords in the tests and exporting the required results to the database, stopping of the timer variable is required now since the execution is completed and test execution time span needs to be recorded. The build is successfully compiled and total compilation and execution time is hence displayed along with the path of generated XML report [14]. These changes are saved to the database now and the results are displayed according to the Boolean state (Passed or Failed). Hence, here ends the keyword routine and connection to the database ends. This completes the test automation with the algorithm of CAF. The graphs formed below are resultant of test specific research where no. of test cases used: 5 and number of functions/keywords used: 38, number of records used as test data: 12, average number of cases used as call to keywords and data (corresponding to each test case): 5. These graphs have been constructed keeping in mind this data. This data has been used again as CAF and D-D comparison statistics.

TABLE 4.1: COMPARISON BETWEEN CAF & DDF

Comparison Parameters	*Automation Frameworks*	
	DDF	**CAF**
Calls On	More calls on the data for a single script	More calls to the script in term of Keywords / methods

Keyword Re-usability	Least focus on keywords , more on data combinations	Keyword utility in functions and functional libraries
Implementation	Easier to the end user because he has to interact with the framework	Easy implementation due to organized scripting and ease in calling Actions and external test data
Accessibility	Somewhat less accessible due to concentration on data alone	Easier to the end user because he has to interact with the framework
User/Customer Interaction	Bit complicated due to interaction with data only	Better due to test segregation and organization
Focus On	Testing entire type (positive, negative) of data values	Script compaction and reusability

Figure 4.1 shows the line graph comparison between the CAF & DDF in terms of data re-usability (the blue line represents CAF method the red line represent the DDF). The CAF graph show less data usage percentage of data for each test case as compared to the DDF with which the data re-usability is more since the framework mainly deals with large amount of data and automation performed with it. This shows the efficiency of DDF.

Figure 4.2 shows the comparison between the CAF & DDF method with the test data redundancy feature (the blue bar represents the CAF method and the red bar represents the DDF). The redundant data percentage as shown above is greater in Data-Driven method as compared to CAF method. This increases the efficiency of CAF method in terms of smart data usage but not in terms of extensive data coverage.

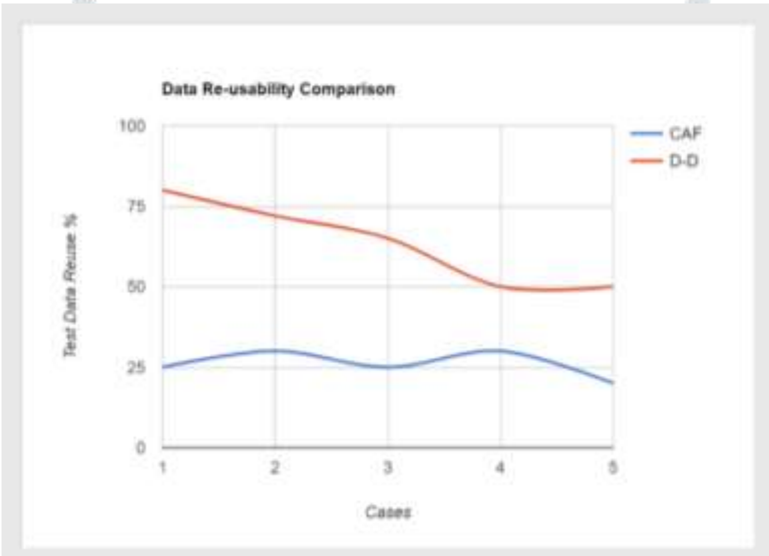

Figure 4.1: Data Reusability comparison curve between CAF & DDF

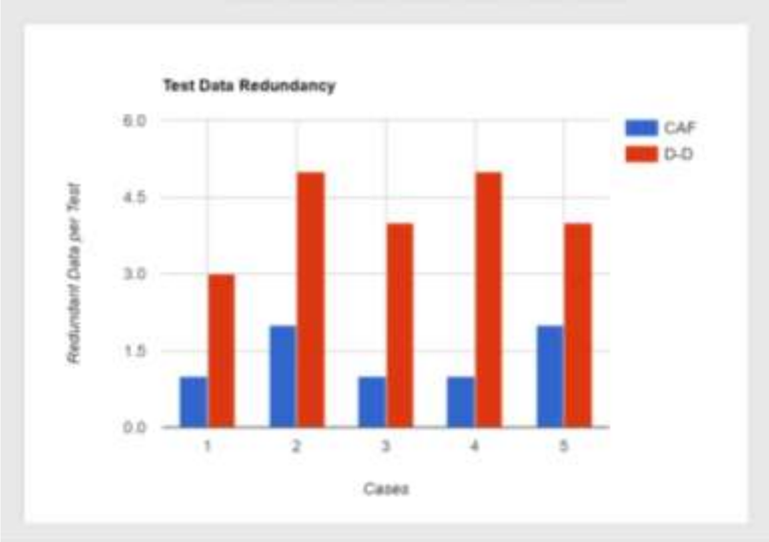

Figure 4.2: Test Data Redundancy comparison between CAF & DDF

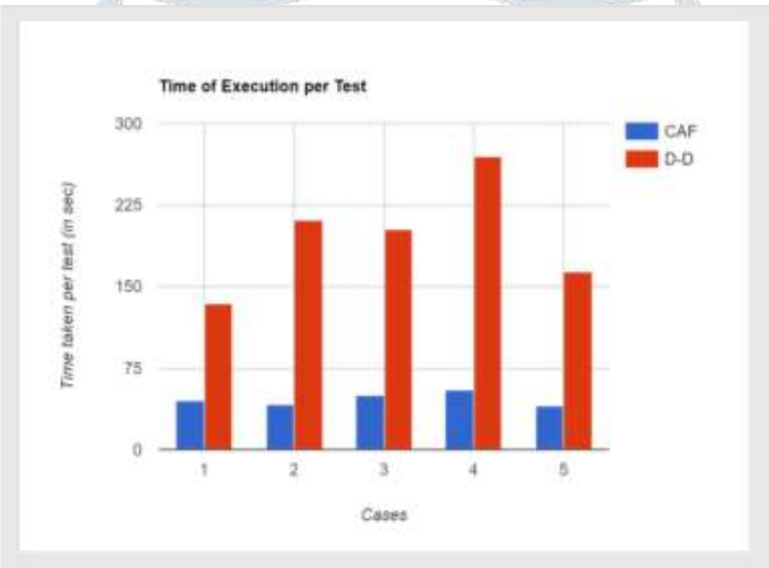

Figure 4.3: Time of Execution per Test comparison graph between CAF & DDF

Figure 4.3 shows the comparison between the CAF & DDF with the time of execution per test feature (the blue bar represents the CAF method and the red bar represents the DDF). The CAF method has greater efficiency in test execution because it takes lesser amount of execution time since the permutation-combinations of data used is less as compared to DDF. The DDF takes the time as a multiple of data (positive, negative) used for each test case. The reference to external data file automatically increases the execution time due to fetching a data numerous time from that database.

While comparing CAF and DDF, it may be analyzed that both the frameworks stand tall in their own approaches. However, in terms of the flexibility of usage CAF has gained the edge. It removed all application specific references and helped the test to become portable and reusable. The CAF is a sub-implementation of the Hybrid Automation framework. It uses the advantage of both Data and Keywords in the test and wins as an all-rounder, if not used in Data centric testing. It has fulfilled the aspirations of a Modular framework with user's interaction with just the Driving-script and encapsulation of rest of the data. The DDF proves to be useful in Data centric automations. Its usage can further be extended to test the Data Analytics or dissect the data warehouses. The DDF is also modular in the sense that one just need to make changes to Data and rest is good to go.

V. Conclusion

CAF approach for selenium web driver is more effective because both data and keywords approaches are combined. When compared with DDF, it is good in terms of reduction in data redundancy and faster execution time. However, the Data-Driven approach works in terms of Data molding and re-usability in the AUT. When the application is more centered on multiple tests for one scenario, DDF works better. Regression and Smoke testing are easier to perform when working with CAF since the modularity of the script is maintained and the efforts to test get reduced. The execution time of CAF approach when compared resulted in better execution time which purely depended on time of recognition of the object on the application and involved less data evaluation. The DDF can also be extended further to XML report generation or defect tool integration system just like CAF. Both frameworks are easy to implement, easy to use, easy to expand, easy to maintain and stimulate. When used to its full capabilities the CAF can be used as platform and technology independent framework having had better speed during automation and reduced investments, better ROI at last. The DDF although not very updated can be used as Extended CAF on using its data handling capabilities along with CAF.

REFERENCES

[1] K. Metha and S. Chavan, "*Towards Developing a Selenium Based GUI Automation Test Pro: a Comparative Study,*" International Journal of Engineering Research & Technology (IJERT), Vol. 3 Issue 2, 2014.

[2] R. Angmo and M. Sharma, "*Performance Evaluation of Web Based Automation Testing Tools,*" The Institute of Electrical and Electronics Engineers (IEEE), 2014.

[3] M. Bolton, "*Rapid Software Testing,*" Agile Conference, Berlin, Germany, 2010.

[4] F. Wang and W. Du, "*A Test Automation Framework Based on WEB,*" 11th International Conference on Computer and Information Science (IEEE/ACIS), 2012.

[5] R. A. Razak and F. R. Fahrurazi, "*Agile Testing with Selenium,*" 5th Malaysian Conference in Software Engineering (MySEC), 2011.

[6] W. D. Yu, G. Patil, "*A Workflow-Based Test Automation Framework for Web Based Systems,"* Computer Engineering Department, San Jose State University, IEEE, 2007.

[7] A. M. F. V. de Castro, G. A. Macedo , E. F. Collins , A. C. Dias-Neto, "*Extension of Selenium RC Tool to Perform Automated Testing with Databases in Web Applications,"* The Institute of Electrical and Electronics Engineers (IEEE) ; San Francisco, CA, USA,2013.

[8] M. Leotta, D. Clerissi, F. Ricca, C. Spadaro, "*Repairing Selenium Test Cases: An Industrial Case Study about Web Page Element Localization,*" Proceedings of the Sixth International Conference on Software Testing, Verification and Validation, IEEE Computer Society Press, 2013.

[9] L. Nagowah and K. Doorgah, "*Improving Test Data Management in Record and Playback Testing Tools,*" Proceedings of the International Conference on Computer & Information Science (ICCIS). IEEE Computer Society, 2012.

[10] N. Uppal and V. Chopra, "Design and Implementation in Selenium IDE with Web Driver", International Journal of Computer Applications, 2012.

[11] P. Yalla, L. SS Reddy, M. Srinivas, T.S.M. Rao, " Framework For Testing Web Applications using Selenium Testing tool with respect to Integration Testing", IJCST, 2011.

[12] R. Angmo and M. Sharma, "Web based Automation Testing and Tools," International Journal of Computer Science and Information Technologies (IJCSIT), 2014.

[13] M. Niranjanamurthy, R A. Kumar, S. Srinivas, RK Manoj, "Research Study on Web Application Testing using Selenium Testing Framework", International Journal of Computer Science and Mobile Computing (*IJCSMC), 2014.*

[14] Oshin and Ashima Arya, "Selenium Testing based on Combined Automation Framework for Web Application", Journal of Advances in Computational Intelligence and Communication Technologies *(JACICT), 2017*.

\