COMPARISON BETWEEN COMBINED AND KEYWORD DRIVEN AUTOMATION FRAMEWORK FOR WEB APPLICATIONS

[1]Oshin, [2]Vineet Chaudhary
[1]M.TECH, [2]M. TECH
[1]Department of Computer Science & Engineering, [2]Department of Computer Science & Engineering
[1]Deenbandhu Chhotu Ram University of Science and Technology (Murthal), Sonepat, INDIA
[2]Maharshi Dayanand University, Rohtak, INDIA

Abstract: Combined Automation Testing Framework (CATF) and Keyword Driven Framework (KDF) are the adaptive frameworks for testing a Web Applications via automation without having a deep coding knowledge. Both the frameworks allow developing a system that can be platform independent functions, amount to faster script execution and good maintainability. This paper presents a comparison of the CATF with a KDF in test automation depending on many comparison parameters. The CATF framework has outweighed the KDF because it combines the advantages of both KDF and Data Driven Testing Framework (DDF). The CATF performs better script utilization, better database repository is maintained, and users interact more at framework level then at coding level. It makes the framework more language and platform generic. A CATF performs like KDF in absence of well-maintained database or a large test data repository.

***Index Terms* - DDF, KDF, TDF, ADF, CATF, HAF**

I. INTRODUCTION

There are numerous platforms like web based, desktop based, Unix based terminal, client server based applications. Any of these applications is referred to as Application Under Test (AUT), an application which is to be tested. There are many open source and COTS tools as well for providing Framework oriented automation of the AUT. Few such tools, mainly for Web based testing are, UFT and Selenium. Automation scripts development requires good coding skills. A test engineer can perform a role similar to a developer when it comes to automating an application. This will require a proper understanding of the underlying application and the tool. In addition, the knowledge of a programming language is also required which proves to be a challenge for the testers [2]. Following are desirable about a test automation framework:

- Regression scripts are executed for the newly deployed build. It is desired that the test cases get executed without putting extra efforts.
- Test execution frequency must increase. It is desired that the tests can be easily created and frequently but seamlessly executed
- More tests should be run very often. This implies necessity of fast execution of tests and creating more tests correspondingly.
- For a category of test, it is difficult to repeatedly perform manually. For example, it is nearly impossible to conduct some performances like stress test without automation.
- A better utilization of resources can take place by allowing the testers to concentrate on more productive and rewarding work by automating the repeating and boring cases.
- Reutilization and consistency of test cases is useful since the basics of a test validation are the comparison of the expected and actual results in every iteration of testing. Test execution of finely created cases can be done in different environments as well keeping in mind the parameters of test portability.
- Reusing the tests from previous releases and projects. This can help developers get the feedback faster and give short time in releasing to the market.

The contribution of the paper is as under:

- The paper details the Table/Keyword Driven Framework (TDF) and CATF.
- Further, the paper presents a comparison of the CATF with a KDF in Selenium WebDriver test automation depending on parameters like: Keyword re-usability, ease of usage, POM Implementation, maintainability, focus area of the two frameworks.

The rest of the paper is organized as under:
Section II presents details of previous related works for various automation framework approaches in web applications. The KDF is explained in section III. The CATF is detailed in Section IV. The performance comparison between CATF & KDF based on various parameters is given in section V. The conclusion takes place in section VI. In the end, there are a list references.

II. RELATED WORK

This section disusses about various automation frameworks and approaches used to improve them. Vaidhehi et. al. talk about the Hybrid framework which they have found in their work. The framework is a combination of a library architecture framework and a data driven architecture. This framework contributes in web application UI testing allowing high level OOP language, easier recovery from GUI changes [1]. There are works which given a review of the various automation tools for implementing different frameworks. These tools are being used in GUI testing of a web application. Different tools were compared for their advantages and limitations by Mehta et. al. [4]. There are some specific web based tools like Selenium which serve the purpose of implementing the test suite via some framework. Various selenium technologies were compared by Angmo et. al. to find the best one on basis of evaluated performance [5]. Bolton et. al have looked for solutions for rapidly changing requirements and testing environment. They have brought forth a rapid testing technique that uses a cyclic technique and re-optimizes the testing to suit customer's requirements. The approach was to make sure everything necessary is done and find methods to speed up the project as a whole [6]. Wang et. al have tried to compare the tools that facilitate web automation testing. They have compared JMeter, a protocol level tool and Selenium, a user level tool. On combining their advantages a new tool can be formed which is an automation cum performance testing tool [7] thus forming a dynamic web testing framework. Selenium is a tool apt for Agile testing is discussed by Razak et. al. There are works related to Selenium being a fit tool for agile testing since it accelerates the test execution process and reduces the total cost. Automation testing with Selenium can make the integration in agile a dream come true with a less input cost [9]. Patil et. al have looked for a Workflow based automation approach which designs a ready to perform framework structure. They proposed that framework is desirable if it interacts with a tester and provides real time feedbacks like a live machine. It must have various plug-ins to facilitate various testing processes. [10]. A tester can interact with a database using some tools and forming them into frameworks as was done by Collins et. al. Selenium is a powerful tool and using its component for creating a database connection and automating it for user's ease is a good idea. The tool worked well in terms of efforts and automation rate [11]. Frequent changes in requirements led to changes in GUI very often. This leads to ill management of the test data. Doorgah et. al. proposed a framework to use technologies like xpath and curl to allow adding, deleting and modifying of the test data very easily. This helped in saving tester's time and project cost [12]. Oshin et. al have worked on creating a Combined framework which combined two or more frameworks for automating the project. The new framework was cost and time effective and allowed easy interaction of the user with the testing tool- Selenium. The framework is balance of methods and data which constitute the entire testing process [17]. Oshin et. al have worked on comparing the Combined and Data driven framework for automating the project. Their conclusion was that Combined framework is appropriate in any testing scenario. Data driven framework was best for data centric testing [18].

2.1 List of Components needed for Testing

2.1.1 **Excel Sheet:** Most of the data needed for interacting with the automation tool is stored here like test cases with their Test Steps, a driverscript to have access to the test cases and sometimes the action/method portfolio.

2.1.2 **Object Repository:** Can be a .properties file to store the properties of the web application objects, this stores the unique information of the object using special locators.

2.1.3 **Action/Method Portfolio:** In TDF, a function file plays an important role, it stores the entire definition of a function which performs the action on the object.

2.1.4 **Data Sheet:** It is a test data repository for storing the input data to a scenario of one or more objects.

2.1.5 **Execution Engine:** The main coded script for controlling all the import and export to and from a Test Excel sheet, an action portfolio and object repository file.

III. TABLE/KEYWORD DRIVEN FRAMEWORK (TDF)

This framework is a collection of huge repository consisting of object properties and creating organized library file of methods/functions. This provides better organization of scripts. In addition, this also enhances readability and accessibility. The set of methods are used as keywords in automation scripts for recognizing the action to be performed on an object. The Keywords are self-guiding in themselves. They give a very clear idea of the operations that need to be performed on the object. The test script structure looks like a tabular structure. This is also called the Table Driven Framework (TDF). We need to make sure that the action words and test data used in TDF must be independent of the automation tool. This allows it to be reusable and portable to any platforms. TDF is implemented for functional and regression testing specifically for the end-to-end flow. This testing can be termed as Action word based testing. The working of the TDF categorizes the Test Case script in five different parts. First is a Test Step which also corresponds to the Manual step in a manual test case, second is Screen/Page to which the Test Object belongs, third is the Object name corresponding with the screen name, fourth is the Action word on Test Object and fifth is the variable which points to the specific data repository of the Test Object.

- *Test Step:* It is the description of the action that particular test step in a test script performs.
- Screen Name: It gives the script designer a chance to uniquely identify the test object.
  Test Object: A Web Page object or element is termed as Test object, for example, Login button, Password text field.
- *Action*: It gives a name to the operation that needs to be performed on the object such as the click, input, open browser.
- *Test Data Variables*: These variables give a pointer to look for the corresponding test data in the Test Data repository.
- *Test Data*: These are the fixed or varying values given to an object.

Table/Keyword driven framework is also called action driven framework (ADF). To implement the tests in this framework, it is not at all important to write a Java code since the code is already separated from the scripts. To create many test cases that can be tested automatically, a table driven framework is fit. Data driven framework is just used to run one automation test script with huge amount of testing data. In practice, they are combined together to make a combined automation testing framework as in (Fig. 3.1). So that one can benefit from both their advantages.

A TDF is very similar to the test script modular framework, but it splits the AUT into methods and actions (or items and methods with regards to the setup language) rather than just the scripts. This kind of framework requires the creation of function library files like generic or application specific libraries that represent modules and methods of the AUT. Those library portfolios can be then called directly from test scripts. Numerous same type of scripts lead can be categorized together and, thus, lead to modularization this framework. This can help in enhancing maintainability of the test scripts. The basic idea of TDF is to find the common set of actions and categorize them into a single action word to perform these actions easily and repeatedly in the operations. Besides, one needs very less technical expertise in operating the TDF because on setting it up, the non-technical testers can easily write the test scripts. It is also easy to understand the framework because one has to write and maintain the code and the action words in the excel sheet itself. On the other hand, maintaining a framework leads to early start of the project because we have everything ready to create a script. We can readily use the object repository, the library portfolio to mirror the manual test processes and the main driving script. We can also benefit from the re-usability of the components by modularizing the code and leading to a less development cost. Maintenance costs can be improved by ensuring the portability of the framework. Since, the TDF has only a single execution system; there is a great requirement for re-usability of the code. However, preparation of the scripts with all its important components can be time consuming and complex. On the other hand, the test data is mostly hard coded which defies the semantics of a good framework.

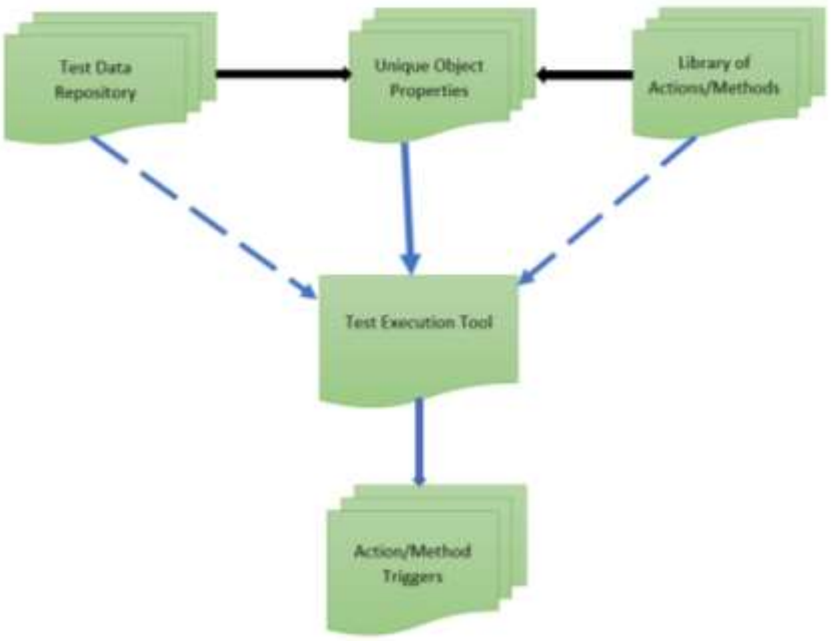

Figure 3.1: Model of Table/Keyword-Driven Framework

IV. COMBINED AUTOMATION TESTING FRAMEWORK (CATF)

This Framework is being combination of function library portfolio and Data driven framework. The powerful utilities and libraries of the library portfolio interact with an external test data source. The added actions/methods inside the library portfolio can have some advantages that make the test data and script less prone to failure. The existing scripts get gradually and manageably converted by using utilities. We develop some common methods that have access to the generic library and can call them in the test script. Alongside, we have the storage media like flat files, SQL, any database which provide test data to interact with the methods [1]. Both table driven and data driven frameworks are integral part of a successful framework. The data driven scripts more compact and less prone to failure by the library file utilities. Table driven capabilities can be achieved using the library utilities that make the scripts manageable and failure prone. A normal framework uses some scripts which become very difficult to implement again in pure table driven approach. It is helpful in effectively managing the hardware resources and the manpower required for testing and releasing the web application.

CATF is made up of many sub-components like: Library portfolio (user specific and application specific), POM (Page Object Model) , Repository for object storage, Action keyword methods, variables & constants, TestNG/Junit reporting mechanism, tracking and logging defect mechanism, Exception handling as in (Fig. 4.1).

Hybrid framework is the parent to the CATF framework. The only difference is that, here, we do not complicate the framework design as in Hybrid Automation Framework (HAF).This framework will still work in compact form. The DDF has its own shortcomings that are reduced by CATF like, need of a good learning curve, having knowledge of the framework's underlying programming language, need for complex processes like establishing connections, fetching parameterized data. Also, the shortcomings of KDF are also addressed by CATF. These include providing better flexibility, more support to data driven architectures, dependency reduction on the specific framework, reducing the time consumption of processing the action/keyword call, good project change handling. By minimizing the coding effort, the automation efficiency is increased. It is developed in a way that it being scripted is replaced with more framework re-usability and less script-ability. Both DDF and KDF along with other features are mainly implemented using external databases like Excel, SQL, CSV files. Any live automation project can successfully use it to its best capability. While speeding up testing the test design, CATF ensures the flexibility so that the framework can interface easily with any automation tool. Lastly, not to forget the reporting capabilities in CATF which are missing in any individual framework approach. This ability allows it to recognize and debug the problem with the test script easily as compared to DDF and KDF. Flow of CATF looks like:

- For the execution of the test script, the excel sheet must have the input. Once the execution is kicked off , the Excel reader file will go through the excel sheet to find out the screen, object name combinations, the actions and respective test data set provided in the scripts.
- Once the test script is visible to the executor, it gets loaded into the ExecuteTest.
- ExecuteTest (along with its library portfolios, reusable libraries, object repository and its unique identifiers) executes the test case.
- The test data is read from and validated at the database storage like excel, SQL or others.
- As soon as tests are executed, the GenerateReport unit comes to work. It take the screenshots of the executed tests and produces a report with pass/fail test steps.
- The results of all tests get stored in a specific folder.[3] and (Fig. 4.2).

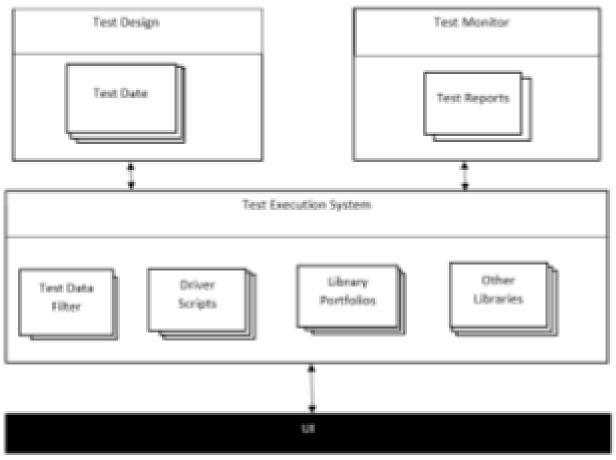

Figure 4.1: Model of Combined Automation Testing Framework (CATF)

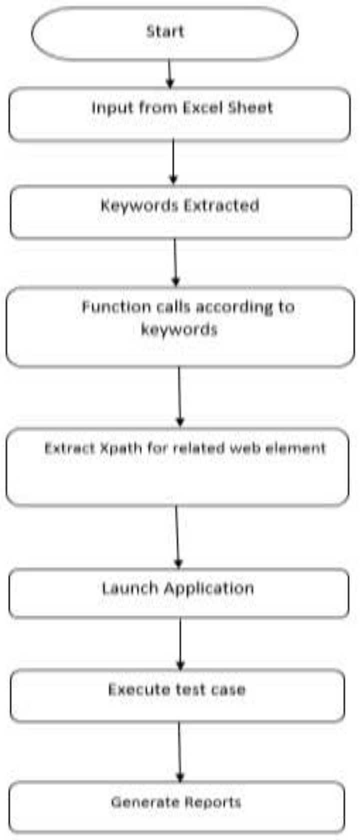

Figure 4.2: Flowchart for Combined Automation Testing Framework (CATF)

V. PERFORMANCE COMPARISON OF CATF WITH KDF

We do the performance comparison of the frameworks to find out which framework performs better in which environment [4][9]. As for any framework, the users are curious to learn the ease of interacting with it. They are least concerned with the internal structure. All they want is ease in designing the scripts, connecting all important utilities to the framework and getting the results generated. CATF can promise this capability better than KDF because it provides more user friendly interfacing. Further, its functionalities can be molded in number of ways without affecting the modules of the framework. This makes CATF more flexible. This further ensures better maintainability. The module interaction may be complex but a single module architecture is simplified. However, without proper documented proofs and artifacts it becomes very difficult for anyone to decipher what it is the framework is designed to do without adequate documentation [6]. Using a well-developed framework like CATF, manual tests can be quickly converted to automation scripts, which reduces coding efforts and time by delivering scripts quickly in regression cycles. Human errors, which occur while modifying complex scripts, are avoided. It filters out the requirements clearly for a tester after doing exhaustive test coverage. Once the framework is deployed, it will take no time in designing and executing a suite of test cases smoothly. The regeneration of cases for regression also becomes easier once properly integrated with test automation/execution tool. This framework can replace the human workforce with automation being just a click away.
The Table Driven framework allows smooth sailing through validations but at some point the tester has to interact with the code. The TDF is dependent on the keywords to be existing in the system. It interacts with data but data is not much focused except for being parameterized. This calls for developing a separate DDF. The ease of interacting also compromised. Below is a comparison table which highlights the advantages of both framework types (Tab. 5.1).

TABLE 5.1: COMPARISON BETWEEN KDF & CATF

Comparison Parameters	***Automation Frameworks***	
	KDF	**CATF**
Calls On	Focus on Action/Keyword call	A compact call which involves internal interaction of the DDF, TDF and other utilities
Keyword Re-usability	Ensures keyword re-usability to its full capability	Keyword utility in form of separately categorized files (generic keywords, application specific keywords)
Ease of usage	Easy when working only with Keyword driven system	Easy from the perspective of a fresh user

POM implementation	Mostly not used	Is used here
Maintainability	Complex maintainability due to keyword changes and adaptation issues	Easy maintainability due to great extent of platform independence
Focus On	Implementing methods by calling actions	Reusing the module to be further modified for minimizing code interference

VI. CONCLUSION

For maximum robustness, we would have to code the state and synchronization tests for every component function call in our scripts. Realistically, we could never afford to do this. It would make the scripts huge, nearly unreadable, and difficult to maintain. Yet, where we forego this extra effort, we increase the possibility of script failure. What we must do is develop a truly application-independent framework for these component functions. This will allow us to implement that extra effort just once, and execute it for every call to any component function. This framework should handle all the details of insuring we have the correct window, verifying the element of interest is in the proper state, doing something with that element, and logging the success or failure of the entire activity. We do this by using CATF, which uses variables and provides application-specific data to our application-independent framework. In essence, we will provide our completed test designs as executable input into our automation framework.

REFERENCES

[1] S. Raju, V. Vaidhehi, "*Design and Implementation of Hybrid Test Automation Framework for Web Based Application"*, International Journal of Innovative Research in Advanced Engineering (IJIRAE), Vol. 4 Issue 3, 2017.

[2] K. Bajaj, "*Hybrid Test Automation Framework for managing Test Data*", International Journal of Pure and Applied Mathematics (IJPAM), Vol. 118 Issue 9, 2018.

[3] B. Bhondokar1, P. Ranawade, S. Jadhav, M. Vibhute "*Hybrid Test Automation Framework for Web Application*", International Journal of Engineering Research & Technology (IJERT), Vol. 4 Issue 4, 2015.

[4] K. Metha, 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.

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

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

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

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

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

[10] 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.

[11] 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.

[12] 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.

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

[14] 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.

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

[16] 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.*

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

[18] O. Rana, V. Chaudhary, "*Comparison between Combined and Data Driven framework for Web Application in Webdriver*", Journal of Emerging Technologies and Innovative Research (JETIR), Vol. 4, Issue 6, 2018.