International Journal of Emerging Engineering Research and Technology
Volume 2, Issue 7, October 2014, PP 51-56
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Data Driven Automation Testing Framework

Prof Vina M Lomte, Rishikesh Chandra[1], Ayush Gondhali[2], Ashish Shinde[3], Sanket Pimple[4]

Department of Computer Engineering, RMD Sinhgad School of Engg, Pune, India
[1,2,3,4]Department of Computer Engineering, RMD Sinhgad School of Engg, Pune, India

Abstract: *Software testing, being a time–consuming and expensive process, there is a great need of finding an efficient solution for speeding it up. Testing Automation would allow us to reduce the cost and will help us in increasing the quality of Software. Automation Framework follows Data Driven Approach which passes data stored in Spreadsheet as input to the number of test scripts written for executing test cases.*

The development of the framework involves the use of Selenium IDE and languages such as Java and JavaScript. Framework automates the testing of software in different phases using a Driver Script and each phase finds the errors in the software and generates a list of errors. Those errors are properly listed in a Spreadsheet and at the same time also contain the code area and line where it appeared, type of error and its possible solutions.

Keywords: *Automation Testing, Selenium IDE, Data Driven Approach, Driver Script, Test Case, Test Script Library,*

1. INTRODUCTION

We all know that the thirst of technology is growing day by day and so is the speed of manufacturing such products and software capable of satisfying the same is also increasing with the same pace. But at the same time we also need the quality assurance for all those products and software so that they can meet the needs and the standards of the clients and the customers which will result in having a long term beneficial and healthy relationship with the customer/clients or end user. And to do the same we need to monitor and test the tool on regular basis either by providing the product lifetime support or by regular testing.

The exponential increase in the growth of Software Industry has brought numerous web applications in the market and these applications are being largely utilized. Hence the correctness of these applications is very much necessary.

Hence testing all these application is mandatory before launching them in the market. But at the same time testing all these applications will require a lot of time and human effort and can also have some probability of human errors. Therefore incorporating Automation in testing is another step demanded by today's need of technological advancement.

The Framework runs a driver script which reads the test cases from a list in a spreadsheet and accordingly selects the testing functions/scripts and concurrently generates a logger report, result report and screenshots of each test case execution and finally generating a complete report on tested application/software, after comparison of the actual result with the expected result.

2. ABOUT THE FRAME WORK

2.1. Steps

2.1.1. Initiation of the Testing Process

Automation Framework will start executing tests with a push of a button and run tests on its own. It can set up the test environment and preferably also check that all preconditions are met. It will read the Test ID from the Spreadsheet and automatically start the testing process.

2.1.2. Starting and Stopping Test Execution

It is possible for us to start the test execution manually. It can be a convenient method because tests can be started automatically at a specified time or after a certain event. Easiest way to start execution at a certain time is making it possible to start test execution manually from command line and using operating system's features for scheduling (cmd in Windows and bash in Linux).

We will have a facility to stop or pause the testing process at any time for making a check. The testing process will stop automatically after generating the Test Report, which will be a combination of Logger Report, Result Report and Screenshots. The link of Test Report will be automatically sent to the Tester and in the Spreadsheet

2.1.3. Handling Errors

Part of running tests unattended is recovering from errors caused by the tested system or the test environment not working as expected. Test framework ought to notice error situations and continue testing without manual intervention.

2.1.4. Verifying Test Result

An integral part of test execution is verifying test results. Framework will automatically verify test result as one or more comparisons between actual outcome of a test and predefined expected outcome. The comparison will be done by referring the Spreadsheet which contains a column containing the data for Expected Result.

2.1.5. Assigning Test Status

After a test is executed and its results are verified, it is expected from the testing framework that it generates a status for the same. If the test was executed without any problems and all comparisons between actual and expected outcomes match the test gets a pass status. In every other case the status is fail. Besides the status every test case should also get a short but descriptive status message. For passed tests this message is normally not that important but with failed tests it can give details about the cause of the problem (e.g. "Calculation failed: expected 3, got 2" or "Initializing test environment failed: Not enough space on disk").

2.2. Algorithm Used

2.2.1. **Algorithm.** *Automated Testing Framework*

Start

```
Open("Spreadsheet", 'read');
Read("Testing_ID()") from Spreadsheet();
Read("Function()") from Spreadsheet();
Search("Function()") in Library();
        Find(Function_Script() from Library());
                If(Function_Script()==Found()) in Library()
Then
                Set Status= 'testing';
                Execute(Function_Script())
                {
                        get(Input_Data() from Spreadsheet()) into Function_Script();
                        get(Expected_Result()) into Function_Script();
                        generate(Screenshot(),Logger(),Report());
Send(Combine(Screenshot(),Logger(),Report()) into Actual_Result();
                }
                If(Actual_Result==Expected_Result)
                Then
                        Status()=="PASS";
```

```
                    Else
                            Status()==“FAIL”;
        End If
            Else
                        Print(“FUNCTION SCRIPT NOT FOUND”);
STOP
```

2.2.2. Algorithm Function Definitions

1. Spreadsheet: The file containing the data to be given as input for the complete operation.
2. Testing_ID(): The column in the spreadsheet which contains the Functions which are to be used for testing.
3. Function(): The name of the testing function. This is the input to the library.
4. Library(): All the functions, their definitions and there scripts are stored here.
5. Found(): The operation which assures that the Function is present in the Library().
6. Function_Script(): It is the definition of the function and working of the function is given in it.
7. Input_Data(): The data present in the Spreadsheet, provided as input to the Function_Script().
8. Expected_Result(): The desired result stored in Spreadsheet.
9. Actual_Result(): The result produced by execution of the Function_Script.
10. Status(): It is is for getting the pass and fail status of the comparison made between actual and expected results.
11. Final_Result: It is the combination of report, screenshot and logger file produces after execution of the function script.
12. Screenshot(): It stores the screenshot of every instance of testing.
13. Logger(): It is a log file generated for each testing instance.
14. Report(): It’s the report file containing the outcome description of testing operation.

2.3. Tool Used

Selenium is an open source automation testing tool for web based applications. It runs directly on browser and supports almost all available browsers such as Firefox, chrome, IE, Opera, Safari etc. It runs on all platforms such as Windows, Linux and Macintosh. It’s a very useful tool for System functional testing and browser compatibility testing.

Selenium is a browser automation tool, commonly used for writing end-to-end tests of web applications. A browser automation tool does exactly what you would expect: automate the control of a browser so that repetitive tasks can be automated.. At a very high level, Selenium is a suite of three tools. The first of these tools, Selenium IDE, is an extension for Firefox that allows users to record and playback tests. The record/playback paradigm can be limiting and isn't suitable for many users, so the second tool in the suite, Selenium Web Driver, provides APIs in a variety of languages to allow for more control and the application of standard software development practices. The final tool, Selenium Grid, makes it possible to use the Selenium APIs to control browser instances distributed over a grid of machines, allowing more tests to run in parallel. Within the project, they are referred to as "IDE", "Web Driver" and "Grid". Selenium is a browser automation tool which lets you automated operations like: type, click, and selection from a drop down of a web page.

3. FRAMEWORK ARCHITECTURE

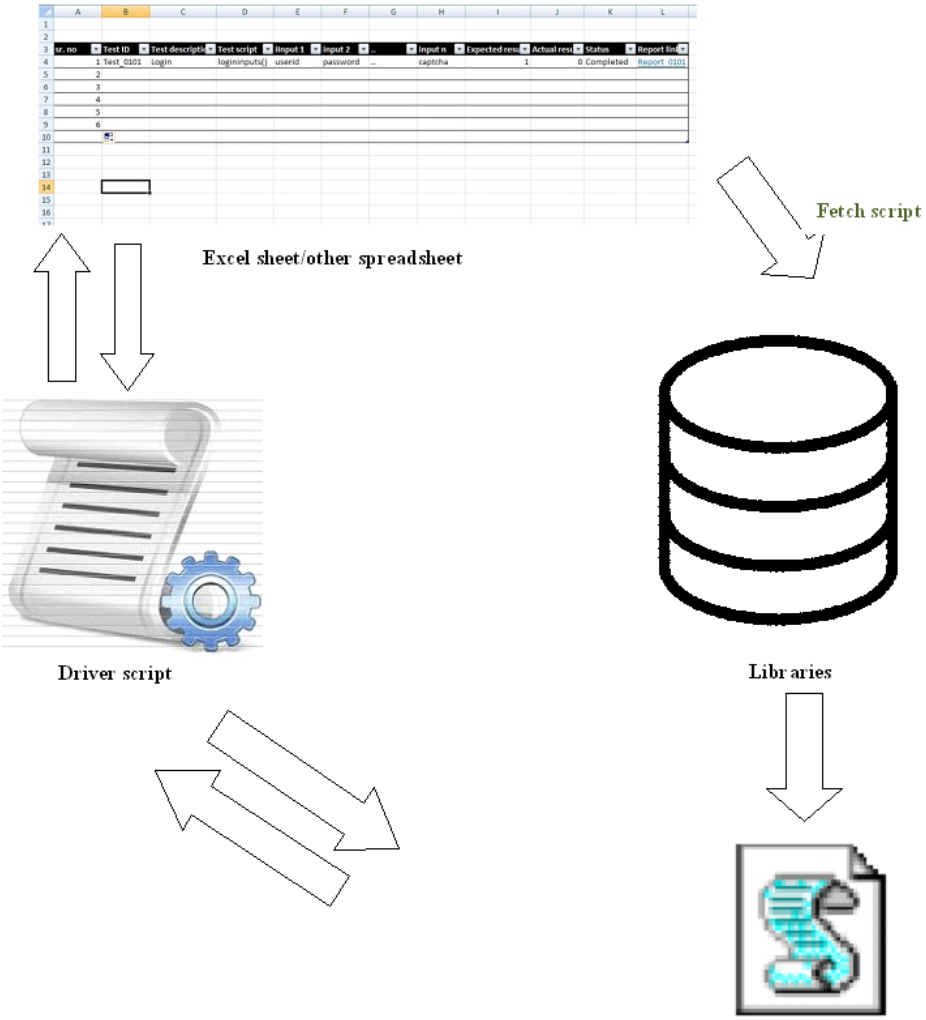

Fig1. *Framework Architecture*

3.1. Here in the Above Diagram the Workingof the Framework is Shown

1. Read the spreadsheet and select the first row.
2. Store all the information about the test suite in the temporary variables.
3. Choose function/test script from the function column and search it in the libraries.
   i. If not found generate error.
   ii. Else, pass the parameters from same row to the function.
4. Execute Function.
   i. Take screen shots and store in reports.
   ii. Store Reports in HTML format
   iii. Create Log files.
   iv. Store report in excel/ sheet/spreadsheet(pass/fail/status)
5. Repeat 2 to 4 for all rows in excel sheet.
6. Generate final report and store it.
7. Generate necessary report links in the spreadsheets.

4. STATE TRANSITION DIAGRAM

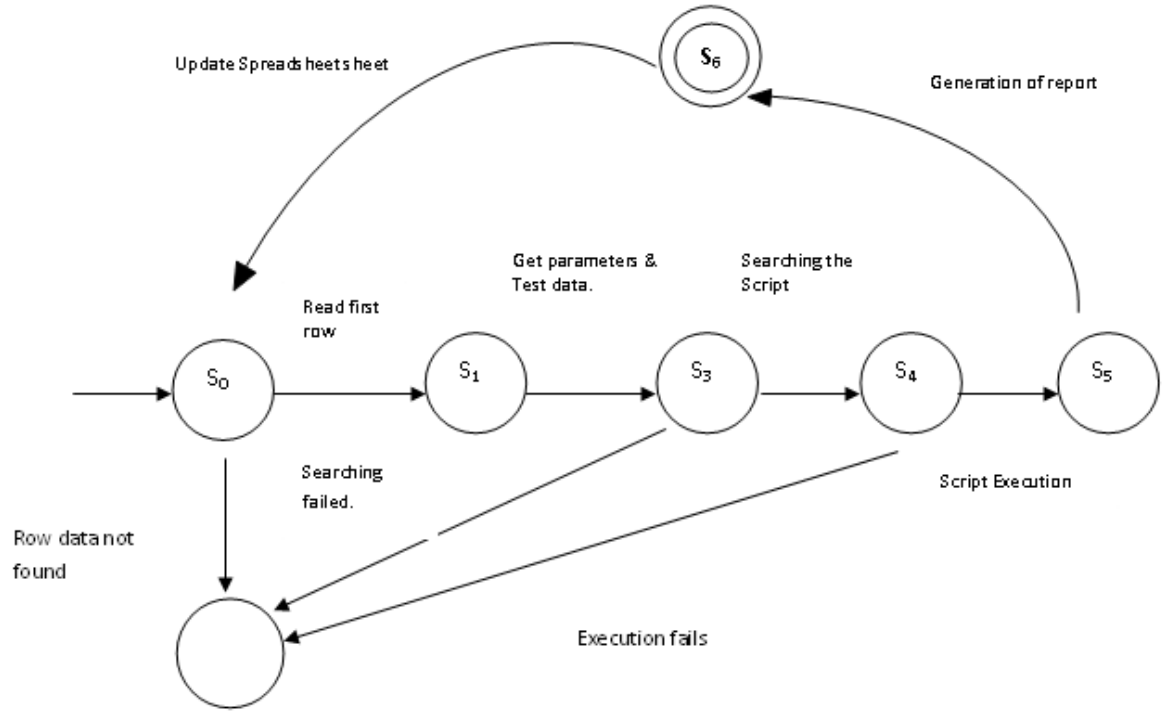

Fig2. *State Transition Diagram*

S_0: The initial state of the system, when the drivers script reads the excel sheet /spreadsheet.

S_1: Read first row.

S_2: The test data is recorded.

S_3: Search the script function column. If found, go to next state, else transition to a dead state.

S_4: Script found

S_5: Execute script. Generate outputs.

S_6: Generate reports. Repeat from S_0.

5. CONCLUSION

A test automation framework can be built around model based testing that begins with automating the test design exercise and the development of manual test procedures. It further uses automation frameworks to attain modularity in automated test script design, and integrates with test execution tools for automated test execution.

Such a framework will help test organizations to:

Greatly scale when test designers and test automation developers are in short supply by training regular testers in model-based testing and test automation

Deliver test automation quickly so that regression cycles can be automated right from the first cycle, and save effort and cost that would otherwise have been expended on manual testing.

REFRENCES

[1] Data Driven and Keyword Driven Test Automation Frameworks by Pekka Laukkanen http://eliga.fi/Thesis-Pekka-Laukkanen.pdf

[2] Wikipedia: Test Automation http://en.wikipedia.org/wiki/Test_automation

[3] NASSCOM Research & Intelligence - Software Testing: Shifting from Functional to Business Assurance

[4] The History of Software Testing by Joris Meerts http://www.testingreferences.com/testinghistory.php

[5] Wikipedia: Requirements Analysis https://en.wikipedia.org/wiki/Requirements_analysis

[6] http://www.hcltech.com/blogs/engineering-and-rd-services/model-based-testing-%E2%80%93-superior-alternative-traditional-software-te

[7] Model-Based Testing (MBT) – HCL Whitepaper by Naveen Jain published on 14 Jun 2013 http://www.hcltech.com/white-papers/engineering-services/model-based-testing

[8] Generations of Test Automation Tools by Mark Utting http://www.cs.waikato.ac.nz/research/

AUTHORS' BIOGRAPHY

Prof Vina M Lomte is an Asst. Professor under Department Of Computer Engineering. She is having 10+ years' experience in the field of teaching as well as research. Her Research interests include domains like Web Security and Software Engineering/ Testing.

Rishikesh Chandra is a student pursuing his B.E Degree under Department Of Computer Engineering from University Of Pune. He is currently involved in the system developing activities for developing algorithm for Framework. Also he is actively working on languages such as Java and JavaScript for making the scripts for the Framework. He is also working in the development of the frontend GUI of the Framework.

Ayush Gondhali is a student pursuing his B.E Degree under Department Of Computer Engineering from University Of Pune. He is presently working on Mathematical Module of the Framework. Also, he is involved in the development of mathematical implementation of the sub algorithms involved in the System. He is actively working on the languages such as Java and JavaScript.

Ashish Shinde is a student pursuing his B.E Degree under Department Of Computer Engineering from University Of Pune. He is currently involved in the development of different Automation Frameworks and Code Coverage Mechanism of the system. He is also involved in the development of testing and automation scripts using languages such as Java and JavaScript

Sanket Pimple is a student pursuing his B.E Degree under Department Of Computer Engineering from University Of Pune. He is presently working on the Mathematical Model and the transitions involved in the model of the Framework. He is also working on the GUI and backend data handling of the Framework.