International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878, Volume-8, Issue-1C, May 2019

Automation of Desktop Applications using Keyword Driven Approach

Divya R, Nandini Prasad K. S

***Abstract*: *In the recent era, the most challenging task to any IT field or a company is to consistently progress and maintain the standards as for quality and also to develop the software systems efficiently. In most of the projects related to software, due to the constraints of time and value, testing is mostly to be neglected. However, manual testing increases time, cost, and is therefore not economical. Software testing primarily involves execution of a program to seek out errors. Effective testing creates top quality code. Automation testing, otherwise called Test Automation, is a point at which product is tested using software by writing scripts. This is a process of automation wherein the manual steps are automated through the software. Manual Testing requires an individual's effort to be spent in front of a computer for careful execution of test scenario steps. Testing scenario that undergo Automation requires Automation tool for the execution of the test cases. The software of automation gives us an added advantage of testing the system by entering the test data, wherein the results of actual are compared with the expected results to generate detailed test reports. Automation Testing focuses on the key area where the test case suites are executed in a quicker and repeated manner when compared with running the test cases manually*.

Index Terms*: *Automation Testing, Manual Testing, Test scenario, test data

I. INTRODUCTION

Software development life cycle involves requirement collection, feasibility study, design, developing the code, verifying or testing and maintainability. In Software development life cycle, testing is a most important step, which is done to make sure that our product is up to the customer need. Testing can be done either manually or through automation. It is impossible to automate all the features. There are certain areas where test cases can be automated. The applications where the user performs actions like either logging in for a form or registering or submitting forms, the work flow of an application, In the area where we can get to see the large variety of users who wants to have the parallel access to the software, all of these can be automated. Also, the Graphical user interface (GUI) items, connectivity with the databases, Communication protocols, network interfaces, stored procedures as per requirement can be automated.

Revised Manuscript Received on May 20, 2019.

Divya R, Department of Information Science and Engineering, Dr. Ambedkar Institute of Technology, Bangalore, India.
E-mail: divyarangashamaiah@gmail.com

Dr. Nandini Prasad K S, Department of Information Science and Engineering, Dr. Ambedkar Institute of Technology, Bangalore, India.
E-mail: iseofficial123@gmail.com

A. The need for automation

We need to automate because it saves time and money. Manual tester spends lot of time and effort with the system and need to cautiously observe the screens of the application, tries different combinations for the same input and forms of usage, verifies the actual outcomes with the behavior that is expected and captures their perceptions. All of these can be replaced with the Automation Testing. This can also completely suppress human error. Since the software development will have different cycles the software, tests may often need to be repeated, so if tests are once created then they can be reused multiple times. Also, they may be used to execute tests that are not possible to carry out by manual testing.

B. Prerequisites to work on automation

Firstly, the suitable automated testing tool is to be selected, then decide which user scenarios to Automate. Test Early and Test Often. Automation Testing can also follow phases similar to the SDLC model, which involves Requirement analysis, scenario development, test case development and testing. Test data needs to be of good quality for example: (a) Data that fits the use case correctly (b) a set of wrong data (c) A set of data that can crash the application (d) Negative test data (e) Boundary value data (f) Outside the boundary on both sides. Always the automated tests must be resistant to changes in the UI. Automation test cases should have a valid title, which are grouped logically. As a result of this, access of test cases will be easier and reader friendly and finally it must be able to launch automated tests in CI/CD pipeline.

C. Goal of the Test Automation

The basic aim of any Test Automation in the industries is to establish software reliability, increase productivity, to find bugs in early stage, regression to ensure no breakage in functionality and to cover performance tests. In order to achieve the Test Automation goals, we need to ensure that script design goals are met. Following are the script design goals : (1) Easy to understand (2) Re-use: The automated test scripts should be written in a generic way, so that they can be reused in other test cases (3) Reliability: Apart from being software wise "reliable", the automated tests do what it is really supposed to test, to give required confidence, making it reliable for program management to take milestone decisions (4) Testability (5) Usability.

Retrieval Number: A10120581C19/19©BEIESP

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

It describes the ease of use of the scripts and its understandability. (6) Maintainability: simple to the level that basic programming knowledge is sufficient.

II. RELATED WORK

In the recent world as of today, there are infinite number of applications in software, which are written as part of web-based applications that run on the browsers. Wide range of tools are available for automation in the current market. Among the tools available, the most commonly known tool is the selenium tool suite. It is a mixture of testing tools related to the automation. The most dominating advantage over other testing tools is that it allows the testers to use the different framework for unlike test cases. The main goal is in finding the most suitable tool in the selenium suite and then with the similar other tools, they are compared for the similar tasks. Selenium is a suite and is not a single tool for the automation. It consists of four kinds of tools: First tool is for the development known as selenium IDE, secondly is the selenium remote control (RC), thirdly is for web drive and last is selenium grid. Selenium is an openly available source tool that is also portable in nature. The Selenium testing framework also supports a variety of programming languages. The projects that require the need for automation can use this Selenium tool. The automation suite is kept flexible which gives an added advantage to the enterprises for testing in case of a design layer using the accelerators [1].

The test suite of the automation can be executed in a frequent and quicker manner, which is worthwhile in terms of the cost and maintenance for the software products. The main objective is to find the procedure for the design of test scripts and the execution and analysis for the scripts that are failed. A predefined framework that is hybrid in nature is defined for the different types of failures. According to the recent survey, it is seen that it is not possible to cover 100% of the test automation with any of the tool available. For efficient performance with respect to execution time, we can see that executing parallelly can also be implemented [2].

The keyword-driven approach that provides an adaptive framework for mapping the test cases for keyword-based with the different ways of test scripts for different kinds of test environments is discussed. There is something called as the keyword-driven technique that is proposed in this paper, that enables the test cases to run consequently as per the requirements. A test case comprises of steps that needs to be followed, data for the test and the result that is actually expected. A novel framework, containing three layers that are test driver layer, automation engine layer, and test execution layer is proposed [3].

As a result of increasing complexity of the automation systems which include software components, it requires continuous and structured approaches to test the systems. In the IT field the development of the automation systems can use TFD- Test-First Development which is promising approach that mainly focuses on automating the tests for the various components of software. On proceeding with this approach, the TFD fulfilled the expectations of Test–Driven automation which covered the component structure that included the aspects for diagnosis, testing and functional requirements. This approach also ensures the enhancement of the process, product quality and hence the project in the field of automation [4].

III. SYSTEM DESIGN AND IMPLEMENTATION

The framework for automation testing is based on level wise. It consists of three layers such as Low level, Module level & High-level layers, which are loosely coupled. The low-level module is designed to use API's to perform the basic actions like Click, Select, etc., These are common for all the controls. For the module level there is a need of the low-level API's for developing the scripts to perform actions on any particular control. In case of High level, it takes care of arranging the module levels in a sequential order to perform the test case in a workflow as per the requirement. The framework supports multiple technologies like python, c#, java, etc., and automation tools like QF-Test, Test Complete, etc., this framework uses a Keyword driven approach, which is used to separate test cases from coding. Methods are written in the scripts that are used to perform actions on the application. The method names here refer to the keywords. To cover scenario keywords must be arranged in the required sequence in a file. This file is the test case and is separated from the actual scripts. This approach gives flexibility in terms for reusability, adaptability, transparency and maintenance.

The algorithm for automation process:

Step 1: Design the test case scenarios by referring the feature and its requirements.

Step 2: Develop the module levels/scripts for the actions to be performed by using the automation tool and arrange them in a required order to create the test case scenario.

Step 3: Execute the created test case. Ensure that verification points are being checked correctly. Check post conditions. Verify the invalid data sets.

Step 4: Run the tests to ensure software/scripts run repeatedly without software failures or crashes.

The software application that is developed using different technologies for example C, C++, C#, Java, requires a tool to automate that is capable of supporting all the technologies and hence should provide a single testing solution. The automation tools that can be selected for these kinds of requirements are Test Complete, QF-Test. Test Complete is an automated testing environment provided by Smart Bear Company for a wide range of desktop, web and mobile application types and technologies. It supports scripting in multiple languages. Test Complete can be selected because a single tool is required which is capable of testing the complete environment of the application software and should be capable of supporting modular scripting. Test Complete is compatible and is a feasible solution for testing a software as it supports testing of different technologies.

Fig. 1 shows the Test Automation Framework which consists of Test Runner, Tools that are used for automation and Test execution reports. Test Runner is a part of automation framework, which supports the following: Execute test cases developed with multiple automation tools at a time to support multiple technology, UI and console mode of execution, archiving and reporting of test results. Python scripting is used to develop Module levels in Test Complete.

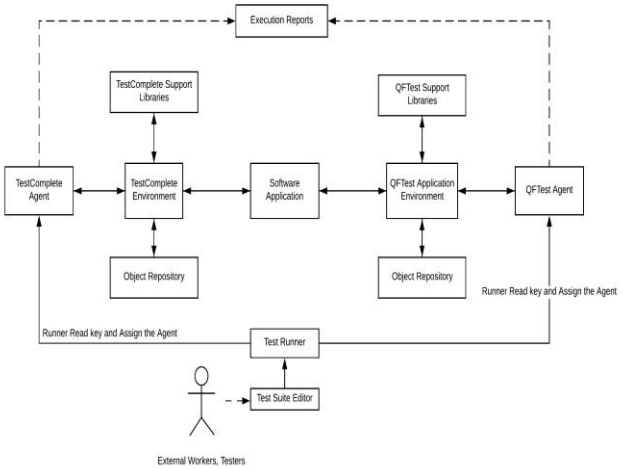

Fig 1. Test Automation Framework

Algorithm to be considered prior to development of test case scenarios:
Step 1: Setup the environment close to the customer deployment.
Step 2: Understand the preconditions and assumptions.
Step 3: Understand the expected behavior.
Step 4: The data to be used must be designed which must include critical values, boundary values, valid and invalid data.
Step 5: Design the test to check the behaviors and states. Clearly identify verification points and post conditions.

A. Developing Scripts

To develop the module levels, the modular scripting methodology is followed. Modular scripting means developing each action on a control as a separate method or script. There are modules where each module and the functionality of that is handled individually or separately which can be achieved by creating the small, individual scripts that symbolize the modules, its sections and functionalities of the test that are given for the particular application. The scripts that are hence created are related to each other in hierarchy in order to build the tests that are lager in size for the better understanding of the specific test case. These modules are written in Python using test complete tool.

The test complete tool looks as shown in the Fig. 2. One can open a new project or existing project and open the required file. The UI to which the module needs to be created is identified and the unique ID of the UI must be obtained. To identify a unique id of the UI, use the spy option provided by test complete as shown in Fig. 3 above.

Click on the spy button, upon which an object spy window appears. There will be two options to spy. Select any one and refer the description to get the properties of the control. Use any of the suitable properties of the control (like AWTComponentName and the JavaFullClassName) and write the method to perform respective action of the control. It's recommended not to use Mapped Name as a property as it was found to leak memory.

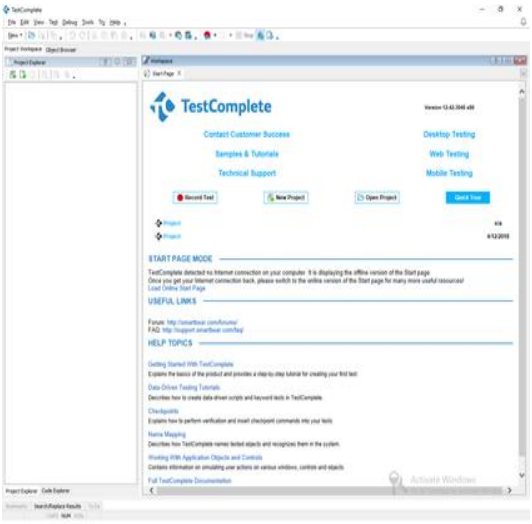

Fig 2. Test Complete Tool

Fig 3. Spy option provided by Test Complete

B. Developing Method in Python

Modules or methods can be developed in Python by using the control properties; class structure should be followed while developing.

The method developed must be unit tested before it can be used, also stress testing (i.e., testing in multiple iterations) must be done to ensure proper working. Once all the required methods are ready, test case files can be developed.
The sample syntax for method is as below:

```
#@ClassName: ClassName
ClassName: ClassName
  #@MethodName: MethodName
  #@Argument1: Argument name to be seen by end user
  #@Argument2: Argument name to be seen by end user
  #@Options: Options if any
  #@Help: Help to use the method if required
  def MethodName (arg1, arg2…):
      .....Required code….
```

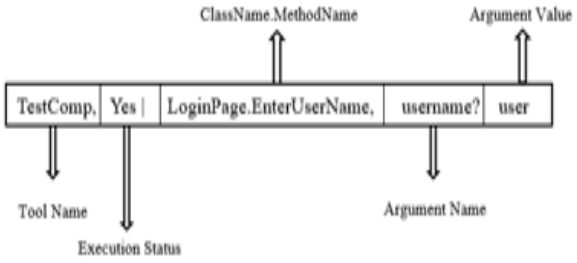

Fig 4. Representation of Test Item

C. Developing Test cases

Test cases are developed by using keyword driven approach. It divides the test case into test item, test step and test case.

Test Item: A test item is a smallest unit of automation framework. It contains an action to be performed on a control.

The Fig. 4 shows test item that represents an action to enter user name. It consists of,

Tool Name - Name of the tool through which the method is written.

Execution Status - It represents whether the test item should be executed. It can have Yes or No value.

Method Call - It represents a call to the method. It must be mentioned as ClassName.MethodName.

Argument Name - It represents the name of the argument.

Argument Value - It represents value to the argument mentioned. In case of multiple arguments, the same syntax for argument name and value should be followed, separated by a comma.

Test Step: A test step might contain one or more Test items. To represent a test step, a hash tag sentence is added, which represents a combined functionality of multiple steps.

For example, in the below sample, the end action of three test items is application log in. Therefore, the three items can be combined in a single step and a name for the test step is given with a hash tag.

#Login to application as user
TestComp,Yes |LoginPage.EnterUserName,username?user
TestComp,Yes|LoginPage.EnterPassword,password?pwd
TestComp,Yes|ClickOnOkButton

Test Case: Test case consists of a combination of multiple test steps which are required to complete the scenario. Every Test case might have precondition and post condition steps as required. In precondition, all the pre requisites for the test case like system settings desired must be done (Optional). In post condition, the changes done during execution must be reverted and should be able to handle any failures and should keep the application in a state ready for next use case execution (Optional).

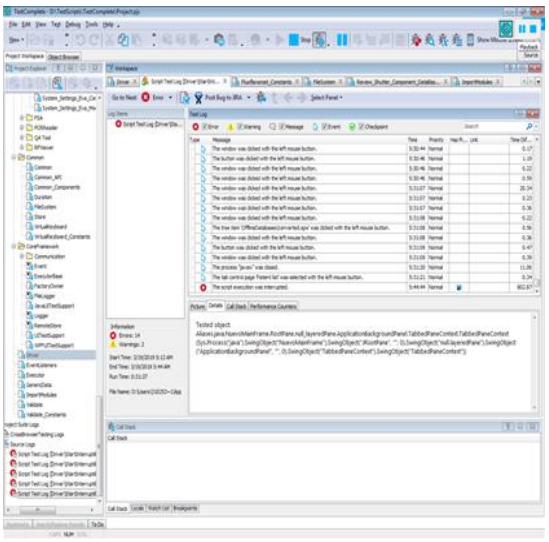

Fig 5. Output screen of Test Complete Tool

D. Writing Keywords

For each test scenario, a Metadata file must be written which should contain the test step name (Keyword) mapped to an Action, Expected, Positive and Negative Results.

For the example of application login, the keyword is mapped as:

Keyword: #Login to application as user
Action: Enter the username and password
Expected: User shall be logged in to application successfully
Positive result: It is possible to login to application as user
Negative result: Login to application as user failed

Once the test case files and the module level scripts are ready, the test case needs to be tested through the runner.

IV. RESULTS AND DISCUSSIONS

After running the automation test cases the test results reports are generated. The results can be viewed in the specific tool itself as shown in Fig 5, which is the log file provided in Test Complete tool. Different tools will have their own form of displaying logs or results. The test log is opened automatically once the test execution is completed. The detailed log is generated which gives the details of each action that is performed, the name of the test, the start time and end time at which the action was performed, the total time taken to run the test, its priority, the information about the errors occurred, number of warnings, indication of pass or fail of particular tests and also the causes of the failed tests etc. It gives summary of test execution.

The test reports are also generated, which will be in the xml form. These reports contain information about whether the test cases are passed or failed. It will also have details of date and time when the test cases were executed. Fig. 6 shows one such test result report.

Fig 6. Test result report

V. CONCLUSION

In Product engineering cycle, testing is the main step. The overview of the automation and manual testing is given in this paper wherein automation has lot more advantage over manual where the effort spent by an individual matter. The automation has several tools available in the market to pick up and to start testing on the application. Testing tools help us to perform regression tests and also, they automate the generation of data to be set up, installation of a product, provides interaction for GUI and defect logging. Tools are specifically selected based on the type of application that needs to test. Hence, particularly specific applications are tested on automation framework. Some of the tools are Test Complete, QF-Test, and Selenium etc. For a tool to be selected it needs to be tested and verified from an application point of view. Since the applications today will be developed using different technologies specifically for desktop applications, the tool that supports for multiple languages is preferred. Because of which the Test Complete tool is focused in this paper. The tool is user friendly and supports modular scripting. The framework supports the execution of test cases written in multiple tools.

ACKNOWLEDGMENT

Authors thank TEQIP-III, Dr. Ambedkar Institute of Technology for providing financial assistance to publish the paper.

REFERENCES

1. Ms. Rigzin Angmo, and Mrs. Monika Sharma, “Performance Evaluation of Web Based Automation Testing Tools” 731978-1-4799-4236-7/14/ @2014 IEEE
2. Hari Sankar Chaini, Dr. Sateesh Kumar Pradhan, “Test Script Execution and Effective Result Analysis in Hybrid Test Automation Framework” 978-1-4673-6911-4/15/$31.00©2015 IEEE
3. Jingfan Tang, Xiaohua Cao and Albert Ma, “Towards Adaptive Framework of Keyword Driven Automation Testing” 978-1-4244-2503-7/08/$20.00 © 2008 IEEE
4. Dietmar Winkler, Reinhard Hametner, Thomas Östreicher and Stefan Biffl, “ A Framework for Automated Testing of Automation Systems” 978-1-4244-6850-8/10/$26.00 ©2010 IEEE.
5. Zeng Wandan, Jiang Ningkang and Zhou Xubo, “ Design and Implementation of a Web Application Automation Testing Framework” 978-0-7695-3745-0/09 $25.00 © 2009 IEEE
6. Hung Q. Nguyen, “The Design and implementation of a flexible, reusable and maintainable Automation Framework”, 2010.
7. Li Feng, Sheng Zhuang, “Action-Driven Automation Test Framework for Graphical User Interface (GUI) Software Testing”, Autotestcon, 2007 IEEE, pp.22-27.
8. Linda G. Hayes, “The Automated Testing Handbook”, page 81-88.
9. H. Kaur, Dr. G. Gupta, “Comparative Study of automation testing tools:selenium, quick test professional and testcomplete,” International Journal of Engineering Research and Application, vol. 3, no. 5, pp. 1739-1743, 2013.
10. https://support.smartbear.com/testcomplete/docs/

AUTHORS PROFILE

Ms. Divya R is currently pursuing M.Tech at department of Information Science and Engineering, Dr.Ambedkar Institute of Technology, Bangalore. She has completed her B.E from SKIT, Bangalore. Her areas of interest include Networking and Machine learning.

Dr.Nandini Prasad K S is working as Professor at department of Information Science and Engineering, Dr.Ambedkar Institute of Technology, Bangalore. She has completed B.E from PESIT, Bangalore; M.Tech from Dr.AIT and Ph.D in Engineering in WSNs. She is a life member in ISTE, ISSS and CRSI. She has authored six books in Engineering for Elsevier and Cengage publications. She has various funding projects from AICTE and KSCST worth around 15 lakhs. She has published and presented sixty plus papers at various national/International conferences and journals. Prof. Nandini is also a Nodal Officer (Academic) in TEQIP-III at Dr.AIT. She has been a reviewer and session chair for IEEE/Elsevier/Springer conferences held in India and abroad. Currently, she is guiding seven Ph.D students. Her area of interest includes Automata theory, Compiler Design, Digital circuit design, Simulation, Big data analytics, Machine learning to mention a few.