RECORDER PLUG IN TOOL FOR ECLIPSE AUTOMATION

Vijaya Dodamani[1], Dr. S Satish Kumar[2]

[1]MTech Scholar, [2]Associate Professor,

Dept of Computer Science and Engineering, RNS Institute of Technology, Bangalore

ABSTRACT

The quality of the software is an important factor in software development life cycle, which should satisfy the user by meeting his/her requirements. The main goal of Automation testing is to uncover problems which are not simply format errors in code but other sorts of errors that are not found by the manual testing during the software development. Different kinds of tools for automate testing have been developed right up until date, but which automation testing tool will be appropriate and sufficient for checking particular software in which period of software development life cycle is not yet clear. Here we are proposing a tool for recording test steps, which helps automating software products that are implemented on eclipse IDE.

Keywords: ***SDLC, Testing, Test automation***

I INTRODUCTION

Every software development group testing its products, yet released software always has defects. Test engineers strive to catch them before the product is released nonetheless they always creep in and they often reappear, even with the best manual tests processes. Test Automation software is the best way to increase the effectiveness, efficiency and coverage of your software testing. Manual software testing is performed with a human sitting in front of a computer, carefully going through application screens, trying various use and input blends, comparing the leads to the expected behavior and recording their observations. Manual tests are repeated often during development cycles for source code changes and other situations like multiple operating environments and hardware configurations.

Quality Attributes

Quality can be measure using the quality attributes. The most common once are discussed here:

Efficiency:

Efficiency is a measure of time required to complete the given task to the system. This can be interns of utilization of processor, disk space and memory.

Usability:

Application should be user friendly. Easy to learn and use. That is , it should be easy for input preparation, operation and interpretation of the output.

Maintainability:

For the different versions of the product it should be easy to add the functionalities to the existing system.

Reusability:

Dividing the application into modules helps for reusing the modules across the application. Hence it makes cost effective and time saving in development.

Reliability:

Reliability measures the correctness of the program under any condition. Application should give consistently correct results for any type of input.

II. LITERATURE SURVEY

The paper [1] titled- “QF-TEST Professional GUI Testing for Java and Web”, gives the customer feed on QF-Test. The difficulties are maintaining tests and updating the evolving component of SUT. More over time is spent on testing rather maintenance, improvements and analyzing what need to be tested.

In this paper [2], “Automated Software Test Data Generation: Direction of Research”- the author defines the test data generation methodologies in three ways. The first one is random test data generation, the second is path oriented test data generation and the last one is global oriented test data generation. These test data generation methodologies are mapped with control flow graph, dependency graph and program dependency graph.

In this paper [3] titled “Improving Quality Using Testing Strategies”, is proposed a scheme to measure quality attributes of software products that need to be achieved and improved by software testing.

III AUTOMATION TESTING

As we know that, main objective of automated testing is to increase the efficiency, effectiveness and to provide robust software for users. From the past few years test automation is being developed to test software product before release. The automation of a software product is being done by writing the test scripts, which helps developer/tester to validate the functionalities that the software provides. So that the automation testing tool can play pre-recorded, predefined test actions which are written as scripts, later the result can be compared with the expected results. Once automated tests are created they can easily be repeated and they can be extended to perform tasks impossible with manual testing.

GUI testing: Graphical User Interface testing provides user to capture and play back the events such as mouse over, mouse click, key press etc, and observes the behavior of the program.

API driven testing: Application Programming Interface provides to ensure the behavior of the program by using APIs.

Test data in system: The input data to be provided for the test script is stored in some storage media in the system. So that, while executing, the tool can fetches the data for the respective test script.

Input data: Input can be any value for the test script that is to check the related functionality. For example, to test the addition of two numbers parameters can be two numbers. Likewise the parameter can be related to the functionality that needs to test.

Expected data: The desired behavior of the system for the given input is Expected data. It can be stored or observed.

Test data fetching: Test data that is stored in the system is read by the tool and feeds it to the test script. This test script passes actual input values.

Reading and executing test scripts: There can be thousands of test scripts for an application but execution can be one at time. So that to store these test scripts one may need central storage for storing them.

Report generation: The report generation provides the screenshot for each script and makes easy to user to analyze the test reports more effectively. It provides human to understand where the functionality of an application is failed to function as defined.

IV PROPOSED MODEL

The automation can be done for any products. It might be web application, android application and many more. Our focus is to provide the automation for the eclipse based projects. The existing tools provide automating eclipse application by writing the scripts in the java language.
One must know java programming language to test any projects that are developed using eclipse. It is time consuming to learn and understand the tool.

To avoid the difficulties faced by the tester for writing the test scripts, we proposed a plug in tool for eclipse, which records the user activities and later those steps can be stored as the test script. The test script can be reused. These recorded steps can be played back to test whether test case steps are correct or not.

The below fig. 1 shows that how the test automation is done by using stored test data. The advantage of our proposed system is that provides excellent reports. From these reports the tester can easily identifies the defect in the system and he can debug it easily.

Automation test architecture

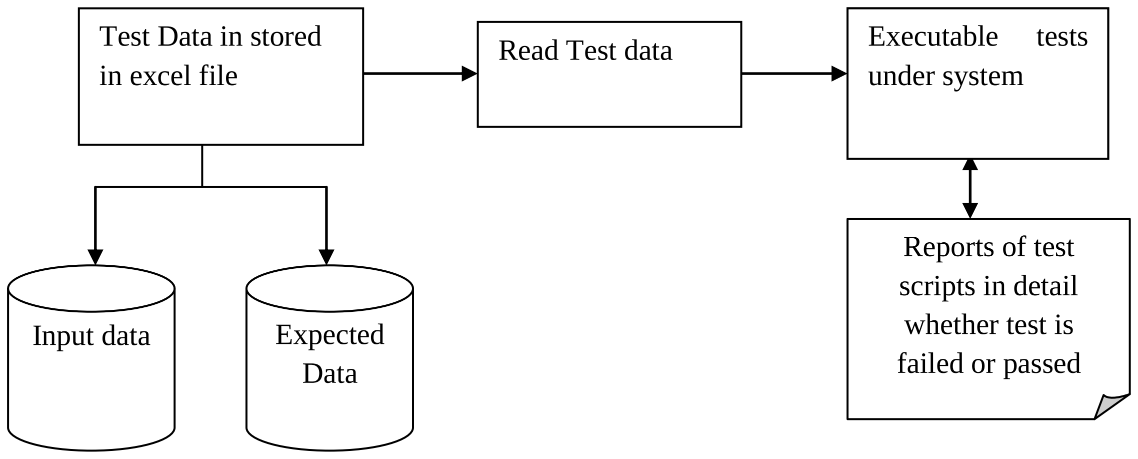
Test Data in stored in excel file
Read Test data
Executable tests under system
Input data
Expected Data
Reports of test scripts in detail whether test is failed or passed

Figure 1 Automation Test Architecture

VI CONCLUSION

Recorder plug-in tool helps user for recording test steps. User can record actions without any difficulties and he can store these steps as tests script. A test script contains several test steps that validates the functionality of the application. It saves time as the automation can happen in parallel with the product development.

REFERENCES

[1] Denis Gauthier, -TEST Professional GUI Testing for Java and Web, 2011 IEEE Conference, July 2010

[2] Hitesh Tahbildar and Bichitra Kalita, Automated Software Test Data Generation: Direction of Research - International Journal of Computer Science & Engineering Survey (IJCSES) Vol.2, No.1, Feb 2011

[3] Sahil Batra, Dr. Rahul Rishi, IMPROVING QUALITY USING TESTING STRATEGIES, Volume 2, No. 6, June 2011 Journal of Global Research in Computer Science

[4] Pretschner, O. Slotosch, H. L¨otzbeyerE. Aiglstorfer, S. Kriebel, *Model Based Testing for Real: The Inhouse Card Case Study*, Proc. 6th Intl. Workshop on Formal Methods for Industrial Critical Systems

[5] Santosh Kumar Swain, Subhendu Kumar Pani, Durga Prasad Mohapatra, *Model Based Object-Oriented Software Testing*, Journal of Theoretical and Applied Information Technology.

[6] Muhammad Shafique, YvanLabiche, *A Systematic Review of Model Based Testing Tool Support*, Carleton University, Technical Report SCE-10-04, May, 2010

[7] Wenfei Fan, AnastasiosKementsietsidis , *Rewriting Regular XPath Queries on XML Views*, University of Edinburgh; FlorisGeerts, Bell Laboratories.

[8] Mikołaj Bojanczyk, Paweł Parys, *XPath evaluation in linear time*

[9] S. R. Dalal, A. Jain, N. Karunanithi, J. M. Leaton, C. M. Lott, G. C. Patton, B. M. Horowitz, Model Based Testing in Practise, Proceedings of ICSE'99, May 1999