International Journal of Engineering Research and General Science Volume 3, Issue 2, March-April, 2015
ISSN 2091-2730

Timed Automation Scripts for Software Testing

Rajeswari C N, Tanya Bala, Darshan Shashidhara

M.Sc. (Comp. Sc.) Student, Department of Computer Science

Christ University, Bangalore, India

bala.tanya @ gmail.com

Abstract— Automation of test plans for testing software is a frequently used technology these days. .The theory behind automation is that it makes the entire testing process much more efficient and requires less manual work. TouchStone is the software under consideration for creating automated scripts. But the drawback still remains that an individual must be present to execute the automated test script and provide necessary variable values. After the termination of the test script, the execution of the next test script must also be manually started. Hence this paper proposes the use of timed runs, where a series of test scripts and their associated variable values will be provided at one go. The start time of the execution can also be provided, if the execution is to be done at a later time. Once the entire batch has run, the test analyst can then review the results, thus saving time and effort.

Keywords— automation; testing; TouchStone; Test Execute; timed automation; efficient;.software;

INTRODUCTION

Every software development group must test its end products, because it may contain flaws. Testing is performed to note the errors before the release of the product and also later for maintenance of the product. Manual testing processes are the most common, but they are less efficient in comparison to automated testing. Automated tests scripts can be created and easily repeated to perform the monotonous tasks which are difficult when performed with manual testing. Automated software testing improves the accuracy of the test plans, as the chance of human errors is removed. It also involves higher test coverage within smaller interval of time, and that too at a low cost. Furthermore if these scripts are run at once, without outside interference, the efficiency increases all the more.

AUTOMATION

Basic Overview

'Automating' means development of automation scripts which can be used for running the manual workflows of the application in an efficient way. In simpler terms it refers to the transfer of human functions to machines.

Test design is one of the foundation phases for software testing. This involves analyzing the requirements and the specifications and synthesizing test cases against which the software will be validated. It helps determine if the software complies with the requirements and specifications. This involves tremendous human effort.

Test development involves developing the test procedures (manual or automated) that will be repeated on the software being tested [2]. Mostly the tests are run manually and hence they involve human effort. Automating software testing involves development of test scripts using scripting languages so that computers can execute these tests with little human intervention. Creation of both manual and automated procedures involves different amounts of effort. However the extra effort required in coding automated test plans is an initial investment that has to be made for the future savings.

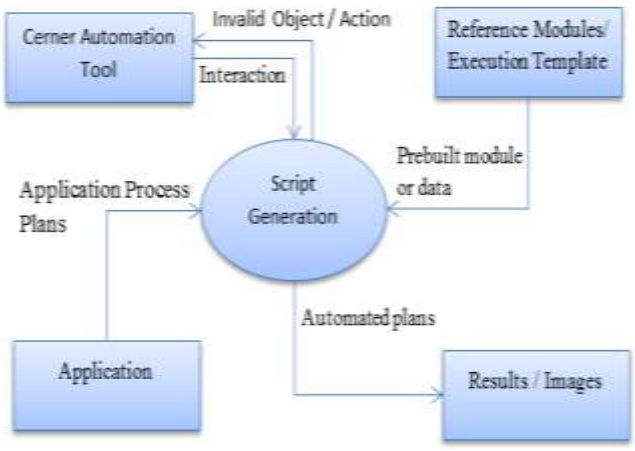

Figure 13 – Block Diagram of Automation

Figure 1 shows the basic components in automated testing. Pre-built modules and data is used along with new modules, and new test scripts are created. The requirements of the test script are obtained from the application to be tested. Automation tools are used to create these test scripts. Once a test script is executed the results can be obtained in the form of an html file or an image.

Advantages

Automation has the following advantages over manual testing :

- Lower cost
- Less human error involved
- More efficient
- Faster than manual testing
- Same modules can be repeated multiple times in the form of a loop
- Reference modules are available providing ease of reuse

Tool Used

The tool used for Automation is an internal tool called TouchStone. It has three basic components:

- Objects –Each component of the software which is to be interacted with is called an Object. It forms a hierarchy, with the root being the application itself.
- Modules – They are units composed of interactions, decisions or validations which can be executed as a whole. Each step in a test plan is represented by a module.
- Interactions –They are nothing but the code for performing a task on the objects

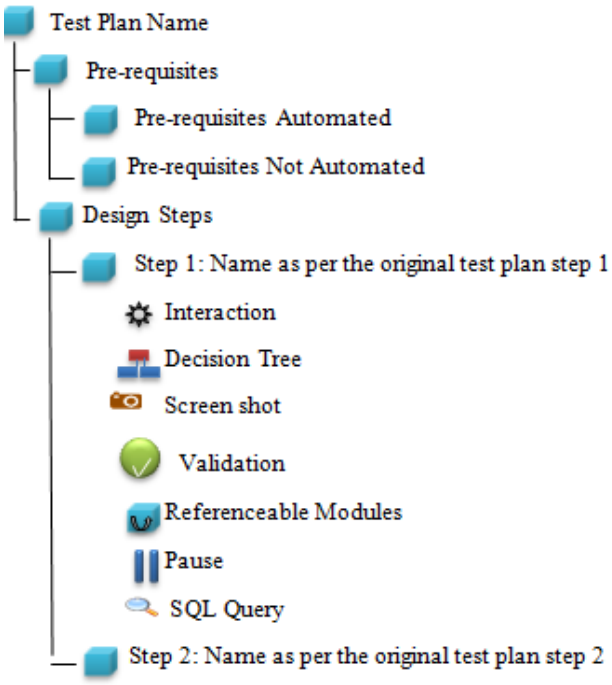

Figure 14 – Parts of a script in TouchStone

The following are other integral parts of scripting an automated test plan :

- Reference Modules – They are reusable codes which are stored in a repository. Using reference modules does away with the need of building code from scratch.
- Execution Templates – Since the test plan is to be run on different domains, with various different settings, execution templates must be used to maintain variable values for one generic automated script.
- Decision Making – When branch paths arrive, decision modules are used.
- Validations – Validations are used to verify whether certain values are being displayed the way they are supposed to be.
- Manual Steps – There are certain steps which cannot be automated, in such situations manual steps are used, where the tester is prompted to perform some action.
- Screenshots – They are used for verification.

Figure 2 shows these basic parts of a test script created using TouchStone and how they are present in a script.

Once a script is created in TouchStone, it can be run as many times as required, but the values for variables must be provided before the script is run. This means that every time a new test plan is to be executed, a Test Analyst must provide the variables and run it. This brings about a waste of resources. That's why timed execution of multiple scripts serially will increase efficiency.

LITERATURE SURVEY

In the paper entitled "Evaluating Automated Functional Testing Tools" [9] Schwaber et. al. showed how testing is a major cost factor in software development. To reduce these costs test automation can be taken as a solution. It has been mentioned that test automation tools can inevitably increase the number of tests being run. The frequency at which these tests are run can also be increased. There is a trade-off between automated and manual testing, and in certain situations manual is better, whereas in other cases automated test plans are much more cost-effective. The paper showed several cost models commonly used to make decisions about automating test plans. What was seen in the paper was that many points must be taken into consideration for automation.

In "Evolutionary testing of classes" [8] Tonella mentions the two common ways of testing software, either manually or automatically. It is shown how the two approaches are complementary in nature. Automated testing performs a greater number of tests in very little time, on the other hand manual testing makes best use of the knowledge of the testing engineer to mainly concentrate on those parts of the system that are known to be prone to errors and faults. The paper proposes a tool called AutoTest that integrates developers test cases into an automated. This combines the advantages of both approaches while keeping a simple interface.

CASE STUDY

There are several regression test plans related to various aspects of the Cerner Millennium application. Let us take an example test plan, EMR-R-Health Maintenance, to see what kind of steps are to be converted from manual to automated. The following test plan will give a general idea of the test plans to be automated.

Description

EMR-R-Health Maintenance deals with testing a specific part of the Cerner Millennium application which deals with providing information related to a patients general health. Once a patient is registered at a healthcare location his/her basic details and problems are charted using the Millennium application. In the health maintenance tab of the application, recommendations are provided for various tests, screens and immunizations that a patient should undergo for better healthcare. These recommendations are made based on the patients' available information which was previously charted. The recommendations are termed as 'Expectations'. Once an Expectation is fulfilled, its details can be charted and it automatically moves to the 'Satisfied Expectations' pane. In this test plan the following functionalities are tested:

- Satisfying an Expectation using a PowerForm – Certain Expectations can be fulfilled by filling in information through a form known as a PowerForm. An example of this form of an Expectation is an 'Alcohol Misuse Screen' which requires the completion of a form having details related to alcohol consumption.
- Satisfying an Expectation using a Procedure – Certain Expectations can be fulfilled by assigning a procedure, such as a 'Blood Test' to be performed on the patient.
- Satisfying an Expectation manually – Some Expectations have an option termed as 'Done' available. These Expectations can be satisfied by simply clicking on the 'Done' option.
- Manually Add Existing Expectations – Not all recommendations are shown at once. New groups of Expectations can be displayed by manually adding existing groups of Expectations.
- Manually Add/Create Custom Free-Text Expectations – Custom Expectations can also be created.
- Change Frequency and /or Due Date – Certain Expectations must be repeated, for example "PET Scan". The frequency of these Expectations can be modified. Some Expectations, like immunizations have fixed due dates. If the user has permission, then he/she can change the due date.

Prerequisites

- The following test user with access to Cerner Millennium application and associated to the testing location must be identified:
    - Registered Nurse – A registered nurse is a clinician having privileges to access the Cerner Millennium application and to chart details for a patient of a specific testing location.
- A testing location must be identified – It is a healthcare organization where the test is to be performed. The registered nurse and patient must be associated with this testing location.
- A new female test patient must be registered (between 50-90 years old) with the following attributes:
    - Active outpatient encounter – An encounter number is associated with a patient every time he/she visits the healthcare center. The patient must be an outpatient, i.e. he/she must not be admitted to the healthcare center overnight.
    - Has an active diagnosis – Any diagnosis such as "diabetes" must be charted for the patient, before the test plan is executed.

Automation Process

The automation process involves conversion of the manual steps of the test plan into a systematic set of modules which automatically interact with the application being tested. The following procedures are to be followed when automating a test plan:

- A root module must be created.
- Each step of the test plan should be taken as a module. These modules must be present as sub-modules under the root module.
- The instructions of each step must be taken as interactions under each module.
- Variables must be created for values which may change, for example username and password
- Decision interactions must be added in situations where branching occurs
- Validation interactions must be used for verification of test steps
- Screenshot interactions must be added where automated verifications are not possible
- After a script is created for the entire test plan, a .xml file is generated which can be used to run the test plan.
- Before the test script is executed, the values for the variables used in the test script must be provided.

TIMED AUTOMATION

Timed automation is nothing but the execution of a set of automated test plans one after the other, without user interference, at a given time. The variable values can be provided beforehand as default values, without manual entry. To understand how this enhances the efficiency we must take into account the time involved in executing scripts. If we have a script with 60 modules, each module having 5 interactions on an average, we will see the following :

- Time for Manual Execution : 50 minutes (taking 10 seconds per interaction approximately)
- Time for Automated Execution :25 minutes (taking 5 seconds per interaction approximately)
- Time for opening a Project and starting execution: 1 minute
- Time for providing variable values: 10 seconds per variable.

From the above data we see that if we want to run 5 test plans with 20 variables and 60 modules each we will get approximate running times as shown in Table1.

TABLE I

Method for Testing	Duration (in minutes)
Manual	272
Automation	147
Timed Automation	130

This shows that timed execution of automated scripts is much more efficient than straightforward automation. With the increase in the size of test plans, and number of test plans being executed, the advantage of timed automation scripts will be more significant.

CONCLUSION

We can see that even though automation of test plans can speed up manual testing, they are still not exhausting all the possible advantages of automation. When automation is timed as opposed to executed manually one test plan at a time, the entire process of testing becomes more efficient. In software companies with multiple products, where a huge number of test plans have to be run on a regular basis, this approach will be very advantageous. Furthermore timed execution of scripts will enhance Overnight testing scenarios

REFERENCES:

1. TouchStone Help Wiki Page : https://wiki.ucern.com/display/associates/ Touchstone+Help+-+deck
2. WHITE PAPER on Test Automation Framework-Dec 2013
3. Mark Fewster,Dorothy Graham, *Software Test Automation*. Great Britain : ACM Press,1991.
4. Mark Fewster,Dorothy Graham, *Experiences of Test Automation: Case Studies of Software Test Automation*. USA : Pearson Education,2012.
5. Automated Testing Process : http://www.tutorialspoint.com /qtp/qtp_test_automation_process.htm
6. Benefits of Automaation Testing : http://www.softwaretesting mentor.com/automation/introduction-to-automation/
7. Marinov and S. Khurshid, "TestEra: A Novel Framework for Automated Testing of Java Programs" in Proc.~16th IEEE International Conference on Automated Software Engineering (ASE), 2001, pp. 22-34
8. P. Tonella, "Evolutionary testing of classes" in International symposium on Software testing and analysis (ISSTA'04). Boston, Massachusetts, USA: ACM Press, 2004, pp. 119-128.
9. Schwaber, C., Gilpin, M., Evaluating Automated Functional Testing Tools, Forrester Research, February 2005
10. M.Grechanik, q. Xie, and Chen Fu, "Maintaining and Evolving GUI- Directed Test Scripts", IC SE'09, I EEE, Vancouver, Canada, 978-1-4244-3452-7, May 16-24, 2009.
11. Myers, Sandler. *The Art of Software Testing, Second Edition*. 2nd ed. New Jersey : John Wiley and Sons Inc, 2004
12. Kaner. *Testing Computer Software.*2nd ed. United States of America : John Wiley and Sons Inc, 1991