World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:9, No:10, 2015

Automating Test Activities: Test Cases Creation, Test Execution, and Test Reporting with Multiple Test Automation Tools

Loke Mun Sei

***Abstract*—**Software testing has become a mandatory process in assuring the software product quality. Hence, test management is needed in order to manage the test activities conducted in the software test life cycle. This paper discusses on the challenges faced in the software test life cycle, and how the test processes and test activities, mainly on test cases creation, test execution, and test reporting is being managed and automated using several test automation tools, i.e. Jira, Robot Framework, and Jenkins.

***Keywords*—**Test automation tools, test case, test execution, test reporting.

I. Introduction

SOFTWARE testing is an important aspect of software quality assurance. It is an evaluation process of a software item to detect differences between the given input and the expected output. The testing result is then used to provide stakeholders with information about the quality of a product or service under test.

The terms "Verification" and "Validation" are frequently used in the software testing world. "Verification" is to ensure that the product is being built according to the requirements and design specifications. It is to ensure that work products meet their specified requirements. "Validation" is to ensure that the product actually meets the user's needs, and that the specifications were correct in the first place. It is to demonstrate that the product fulfills its intended use when placed in its intended environment [1].

In this paper, there are three main software testing activities discussed, mainly on Test Planning, Test Execution, and Test Reporting. Test planning involves scheduling and estimating the system testing process, establishing process standards and describing the tests that should be carried out [2]. Once the test plan is finalized, test engineers may proceed with test design and test cases creation. Test execution includes the execution of test cases or test scripts, manually or in an automated way [3]. Test reporting is to communicate the test results and findings to the project stakeholders so that decisions can be made for the software release [4].

II. The Challenges

The following are the analysis of the challenges faced in the three main software testing activities:

A. Resource and Effort Estimation

It estimates the resources and time required for all the test activities performed during Software Test Life Cycle irrespective of the size of the testing task. The calculation of test estimation can be based on past experiences or past data, documents or knowledge available, assumption and risks. It is always difficult to have an accurate estimation especially when the testing task required some business domain knowledge and the test engineers are new to the business domain and often it requires some time buffering for learning curve before they can be involved in the testing activities.

If there are automation scripts or automated test cases in place, the new test engineer can just trigger the test automation execution and get the test result with minimum business domain knowledge required. This will eliminate the learning curve time and indirectly makes the effort estimation becomes more accurate.

B. Lack of Skilled Test Engineers

The testing in general often involves manual tasks. The test engineer creates test cases and executes the test cases manually, step by step and indicates whether a particular step was accomplished successfully or whether it failed. It is very much depends on the individual's domain knowledge and testing skill. One test engineer may approach and perform a test differently than another, thus, operating personnel human error can occur easily if it is all done manually. Besides, poor domain knowledge of test engineers can end up in ineffective test scenarios, test scripts and post implementation defects.

There was a quote from Elizabeth Hendrickson, "Most good testers have some measure of technical skill such as system administration, databases, networks, etc. that lends itself to gray box testing". A good test engineer should have both technical and non-technical skills. The better understanding of the application is, the better the bugs raised will be. If a test engineer can understand what a stack trace is and why it is happening, the more effective he/she will be in communicating what has happened and why to the developers.

C. Lengthy Test Execution and Reporting Time

During the test execution phase, there might be several test iterations involved. Test engineers execute the test cases and report the test result manually. If the test result is failed, developers will fix the reported issue and test engineers will then retest and verify the same test cases. It is time consuming

Loke Mun Sei is with the Corporate Customer Quality & Strategic Initiatives department of MIMOS Berhad, Kuala Lumpur, Malaysia (e-mail: ms.loke@mimos.my).

if the same test cases were to be manually executed in each and every of the test iterations and also during the regression test phase.

Since the allocated time for testing is limited and the mobile application needs to be published to the market within budget and time, it is usually impossible for test engineers to retest all of the existing test cases. The usual workaround would be to prioritize and to select and test a small subset of the existing test cases based on the timeline available. As a result, the test coverage would be reduced and the risk of defects escaping during the test phase could be alarming and very high.

III. The Solution

Test automation with the objective to address the challenges mentioned earlier while maintaining the quality of the product or service is needed. Hence, we use combination of several automation tools to help automating our test processes.

A. Test Cases Creation: Automate Using Robot Framework

Robot Framework is a generic, application and technology independent framework. The test data is in simple, easy-to-edit tabular format. When Robot Framework is started, it processes the test data, executes test cases and generates logs and reports. The core framework does not know anything about the target under test, and the interaction with it is handled by test libraries [5].

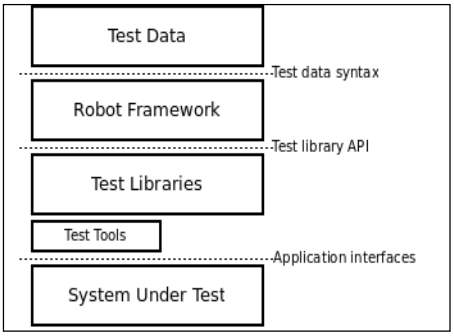

Fig. 1 Robot Framework Architecture

Robot Framework supports both internal and external libraries that can be used to automate test cases cover for the front-end GUI web application (Selenium), back-end testing (SSH, Database), Windows application (AutoIt), mobile application (Android, IOS), etc.

The following example demonstrates how the manual test case created in Jira is automated using Robot Framework script.

- Fig. 2 shows the manual test case for user login steps (Procedures) and its expected result (Expected Outcome)
- Fig. 3 shows the automated test case using Robot Framework script based on the test steps and expected result

Robot Framework utilizes the keyword-driven testing approach and the automation script is written in spaces/tabs separated plain text format. The number of spaces used as separator can vary, as long as there are at least two spaces (or use tab), and it is thus possible to align the data nicely.

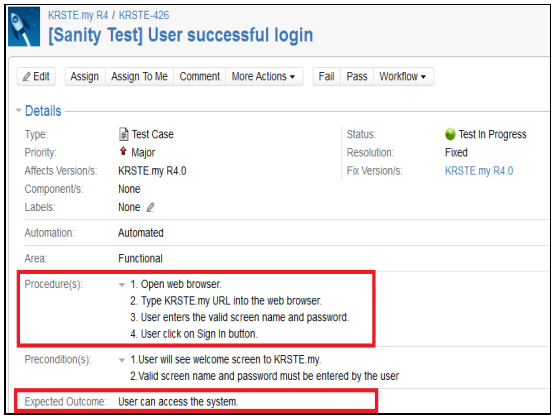

Fig. 2 Manual Test Case

```
*** Settings ***
Documentation  User successful login

*** Variables ***
${URL_TO_BROWSE}  [redacted]
${username}       [redacted]
${password}       [redacted]

*** Keywords ***

*** Test Cases ***
Login to Web
    Open Browser  ${URL_TO_BROWSE}  ${BROWSER}
    Set Selenium Timeout   180
    Click Link    Welcome to KRSTE Release 4, Please Click Here
    Click Link    open   dont_wait
    Input Text    _username  ${username}
    Input Text    _password  ${password}
    Click Button  submit
    Page Should Contain   Welcome,

Logout from Web
    Click Link    css=a[title="Click Here To Logout "]
    [Teardown]  Close Browser
```

Fig. 3 Robot Framework Automation Script

Testing capabilities of robot Framework can be extended by test libraries and users can create own keywords using the same syntax as per the test automation script syntax.

```
*** Settings ***
Library   SeleniumLibrary
Library   OperatingSystem

*** Variables ***
${URL_TO_BROWSE}  [redacted]
${username}       [redacted]
${password}       [redacted]

*** Keywords ***

User Login
    Open Browser  ${URL_TO_BROWSE}  ${BROWSER}
    Set Selenium Timeout   60
    Click Link    Welcome to KRSTE Release 4, Please Click Here
    Click Link    open   dont_wait
    Input Text    _username  ${username}
    Input Text    _password  ${password}
    Click Button  submit

User Logout
    Click Link    css=a[title="Click Here To Logout "]
```

Fig. 4 Sample Own Keywords Developed

B. Test Execution: Trigger Using Jenkins

Jenkins is an application that monitors executions of repeated jobs, such as building a software project or jobs run by cron. Among those things, current Jenkins focuses on the following two jobs [6]:

- Building/testing software projects continuously. It provides an easy-to-use continuous integration system, making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. The automated, continuous build increases the productivity.
- Monitoring executions of externally-run jobs, such as cron jobs and procmail jobs, even those that are run on a remote machine. Jenkins keeps those outputs and makes it easy for notification when something goes wrong.

Once the Robot Framework test scripts or automated test cases are created, we can then trigger and execute the test using Jenkins automatically. Before the test automation script can be triggered, a Jenkins job needs to be created. The Jenkins job consists of the configurations/settings and commands to run the test automation scripts. Sample Jenkins job that is ready to be triggered is as shown in Fig. 5.

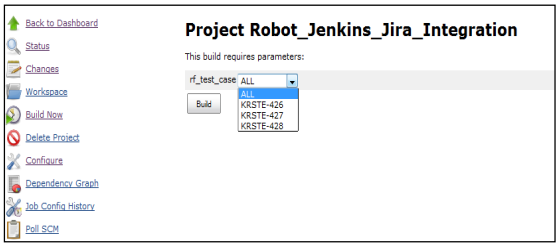

Fig. 5 Sample Jenkins Job to Trigger Test Execution

A test engineer just need to select the test case (select all test cases or a particular test case) and click on the "Build" button and the test case(s) will be executed automatically.

Jenkins job can also be scheduled to run automatically. It can be done by setting up the cron job in Jenkins using "Build periodically" feature. Jenkins Cron job format is in Unix-like computer operating systems as shown in Fig. 6.

Field name	Mandatory?	Allowed values	Allowed special characters	Remarks
Minutes	Yes	0-59	* / , -	-
Hours	Yes	0-23	* / , -	-
Day of month	Yes	1-31	* / , - ? L W	-
Month	Yes	1-12 or JAN-DEC	* / , -	-
Day of week	Yes	0-6 or SUN-SAT	* / , - ? L #	-
Year	No	1970–2099	* / , -	This field is not supported in standard/default implementations.

Fig. 6 Jenkins Cron Job Format [9]

C. Test Reporting – Test Case Execution Detailed Logs: Using Robot Framework Plugin

Robot Framework plugin collects and publishes Robot Framework test results in Jenkins [8]. Fig. 7 shows the sample plugin configuration.

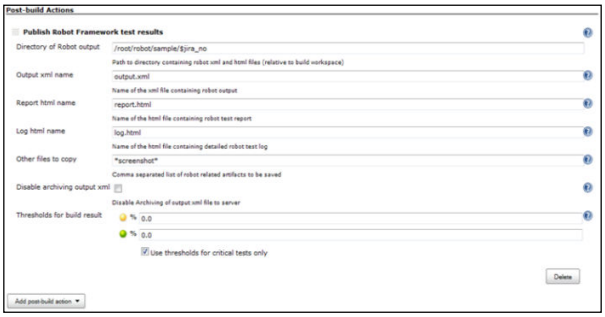

Fig. 7 Sample Robot Framework Plugin Configuration

With Robot Framework plugin, a summary report file and a detailed test case execution log is available after each test case execution. A summary report file contains an overview of the test execution results in HTML format. They have statistics based on tags, executed test suites, and a list of all executed test cases.

When both reports and logs are generated, the report has a link to the log file (log.html) for easy navigation to more detailed information.

The log file contains details about the executed test cases in HTML format. They have a hierarchical structure showing test suite, test case, and keyword details. The log file contains a detailed status on each of the test step whether it is failed or passed. Its detailed status will then help in doing debugging or investigation later. Even though log files also have statistics, reports are better for getting a higher-level overview as in the summary report file.

The sample summary report file and detailed test case execution log file is shown in Fig. 8 (for failed test case), Fig. 9 (for passed test case), and Fig. 10 respectively.

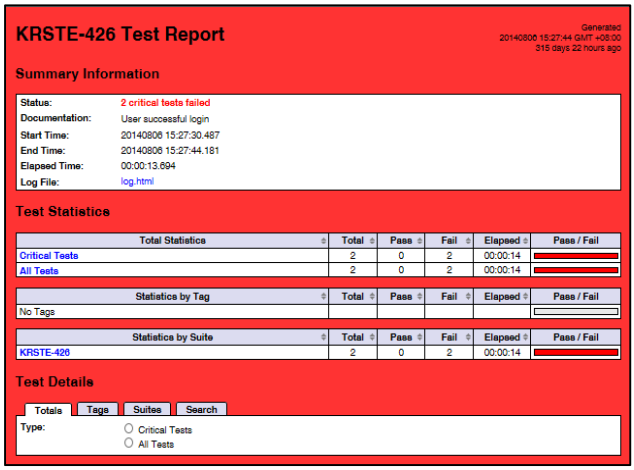

Fig. 8 Sample Summary Report File (Failed Test Case)

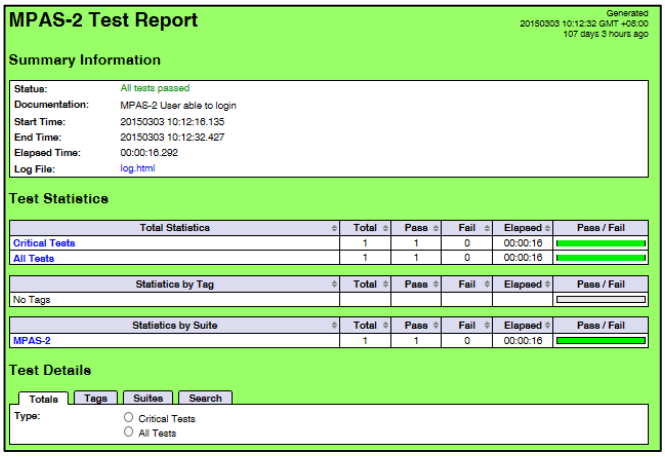

Fig. 9 Sample Summary Report File (Passed Test Case)

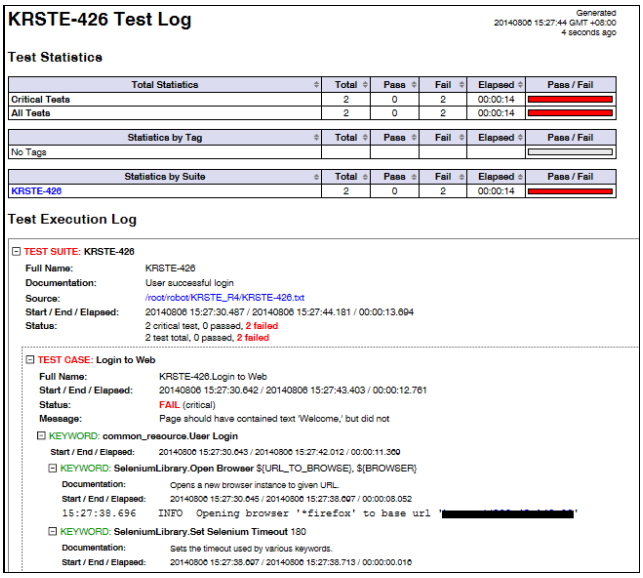

Fig. 10 Sample Detailed Test Case Execution Log

D. Test Reporting – Test Case Status: Update Using Jenkins Jira Issue Updater Plugin

Jenkins Jira Issue Updater Plugin is a Jenkins plugin which updates issues in Atlassian Jira by changing their status and adding a comment as part of a Jenkins job [7]. Fig. 11 shows the sample plugin configuration.

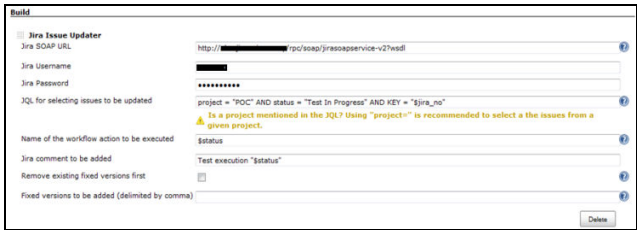

Fig. 11 Sample Jenkins Jira Issue Updater Plugin Configuration

When the Jenkins job is triggered, it will update the Jira status automatically based on the test execution result, as shown in Fig. 12:

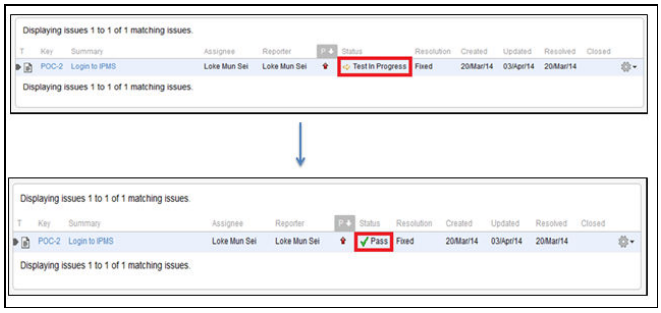

Fig. 12 Jira Status Updated Automatically

IV. Conclusion and Future Works

From the study and research, it clearly shows that the test automation has a great impact on the test activities conducted.

- Resource and effort estimation: Since the software testing activities are now automated, it can then eliminate the test engineers learning curve time and indirectly makes the test effort estimation becomes more accurate.
- Lack of skilled test engineers: Minimum business domain knowledge is required to run the automated test scripts or test cases and thus minimize the personnel human error.
- Lengthy test execution and reporting time: Test execution and reporting is no longer done manually and thus it shortens the test cycle time.

For future work, we plan to integrate with development team so that when development team has changed the codes, it will trigger the automation job to build the changed component, deploy the newly build component to the test server, and then run the existing test cases automatically and publish the result. This could be beneficial to both the development and test team to verify which build is working and which is not, and it is also easier to track and to debug on the broken build.

References

[1] STF, "Verification and Validation: Definition, Differences, Details" in http://softwaretestingfundamentals.com/verification-vs-validation/, August 11, 2011
[2] Ian Sommerville, "Test Planning" in http://ifs.host.cs.st-andrews.ac.uk/Books/SE9/Web/Testing/Planning.html, 2008
[3] Capture Plc., "Test Design & Execution" in http://capture.hu/services/Test-Design-Execution/46/, 2010
[4] Ioan Mihnea IACOB and Radu CONSTANTINESCU, "Testing: First Step Towards Software Quality" in JAQM, vol. 3, No. 3, 2008, pp. 3
[5] Python Software Foundation, "Robot Framework" in https://code.google.com/p/robotframework/, 1990
[6] Kohsuke Kawaguchi, "Jenkins" in http://jenkins-ci.org/, 2013
[7] Laszlo Miklosik, "Jenkins Jira Issue Updater Plugin", in https://github.com/jenkinsci/jira-issue-updater-plugin, 2011
[8] Rishab Jain C and Rajesh Kaluri, "Design of Automation Scripts Execution Application for Selenium Webdriver and TestNG Framework" in ARPN Journal of Engineering and Applied Sciences, VOL. 10, NO. 6, APRIL 2015; pp. 2
[9] Rajesh Kumar, "Setting up cron job in Jenkins" in http://www.scmgalaxy.com/index.php?option=com_k2&view=item&id=894:setting-up-the-cron-jobs-in-jenkins-using-build-periodically-scheduling-the-jenins-job&Itemid=120, 2014