ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 2 Issue Special 1 Jan-March 2014
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)

An Efficient Framework for Unified Automation Testing: A Case Study on Software Industry

[I]A.Divya, [II]Mrs.S.Devi Mahalakshmi
[I,II]Mepco Schlenk Engineering College, Sivakasi, Tamil Nadu, India

Abstract
Manual software testing becomes difficult, time consuming and costly as software systems evolve since it is performed by a human sitting in front of a computer carefully going through application screens, trying various usage and input combinations, comparing the results to the expected behaviour and recording their observations. Manual tests are repeated often during the development cycle for source code changes and othersituations like multiple operating environments and hardware configurations. Therefore tests should be automated to reduce testing efforts. Automated software testing is like writing the code to test the source code, which performs predefined actions, compares the results to the expected behaviour and report the success or failure of these tests. In this paper, automated tests are developed and extended to perform tasks impossible with manual testing in very less amount of time. In order to make automated testing effective and efficient, so that it can reduce overall costs, new techniques and methodologies were applied during the project like Modular Keyword Framework and hybrid approach. It is observed that automated software testing is an essential component of successful development project.

Keywords
Modular Keyword Framework, Manual Tests, Automated software testing, Hybrid approach, Automated Tests

I. Introduction

Every software product needs to be tested adequately, but as quickly and thoroughly as possible with minimal resources. To accomplish this goal automated testing is used.

Faced with this reality and realising that a big number of test cases cannot be executed manually (For e.g. Simulating 1,000 virtual users for volume testing), automated testing is introduced [1], [2]. In the environment of continual changes and additions to the software through each software build in rapid application development, where requirements are encouraged to evolve, software testing takes on an iterative nature itself. Each build is accompanied by a considerable number of new tests as well as rework to existing test scripts, just as there is rework on previously released software modules. Given the continual changes and additions to software applications, automated software testing becomes an important control mechanism to ensure accuracy and stability of the software through each build.

A test development effort may be as timeconsuming as effort required developing the software application. Much of the test effort required on a project now needs to be supported by automated test tools. Manual testing is labour intensive and error prone. And it does not support the same kind of quality checks that are possible with automation testing. Automation testing can replace manual test activities with a more efficient and repeatable automated test environment.

Automated test capabilities for software products include testing of the graphical user interface, requirements compliance, load performance, database validations, etc. New capabilities continue to be added to keep pace with the growing demand of test support. In regression testing which is a risk reduction exercise, it is needed to find the bugs in the code as fast as possible with minimum effort. This can be achieved using automation testing [3].

Automated Software Testing is the best way to increase the effectiveness, efficiency and coverage of the software testing. Once automated tests are created they can easily be repeated and they can be extended to perform tasks impossible with manual testing. Because of this, it is found that automated software testing is an essential component of successful development projects.

II. The Proposed System

A. Design Goals

Automation Testing has found its place in the software industry, with the crucial role that it plays in quality software production. As business requirements grow, so does the pressure to deliver the product with fewer resources, in reduced time and with high quality.

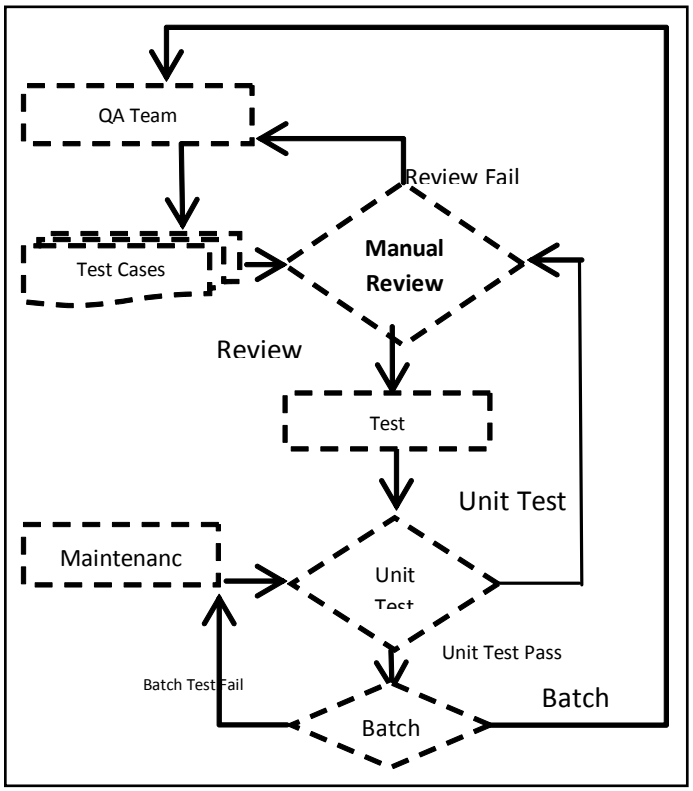

Fig. 1: General Flow of test automation

B. General Flow of test automation

Based on the product/application/module requirement, types of testing that need to be performed are identified. Testing requirements and their nature are studied for the application.

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Each requirement has its own actions, validations for testing. For example, contribute

- Scenario 1: For an application, form validation functionalities, database validation and accessibility functionalities needs to be validated.
- Scenario 2: For an application, all the web-service methods need to be validated. This would also include the delay time for the request's response from third party systems.

Appropriate testing tool is selected which is best suited for the AUT. Every tool has its own limitations. A feasibility study needs to be conducted for the requirements against the tools. This study would result in deciding the testing tool [4]. Test Automation framework is developed which is to be used for developing and running automated tests.QA team which is responsible for management of all Quality Assurance functions writes test cases and does implementation of test process. HP-QC is used as a repository for documenting these test cases. These well documented test cases are then manually reviewed by test automation team step by step. If the test case is manually passed, it is automated. If manual review is failed (i.e. the application is not behaving the way it is supposed to according to steps in the test case in HP-QC), it is reported back to QA team since it can be a bug in the application. Automating test case is nothing but writing a code to test the code. Automated tests are then unit tested. If unit test passed, the test case goes for batch testing. If unit test failed, it is again modified so that the execution of the automated test matches the manually reviewed steps of the test case. All automated tests of the same functionality are batched and integration testing is done. The automated test cases which are failed in batch testing goes under maintenance and are corrected. Theses test cases again are unit tested and batch tested. Once the batch is passed, all these automated test cases are delivered to QA team, so that these can be used for regression testing reducing the manual testing effort.

C. Design of test automation framework

Generally, testing is started with creating test scripts based on the scenarios. This includes multiple actions to be performed against each object. This approach leads to an ad-hoc test script creation and duplicate testing effort, i.e. testers, would create test scripts for a single action in different scenarios.

There is an approach that takes a different path as explained below. For designing a framework, various elements need to be taken into consideration. Utilities/Components (re-usable) would be designed for the following elements that include:

Actions to be performed - Identification of actions to be automated for each object of the application and thus identify and abstract common functions used across multiple test scripts.

- Database Communication - Database validation and check point validations
- Data retrieval - Retrieval of data from multiple input data stores
- Error Handlers - Error handlers to handle known and unknown errors and log the information
- Custom Messages - Display of relevant defined messages
- Result Presentation - Customized and presentable reports on completion of test execution
- Easy to expand, maintain, and perpetuate.
- Decouple test data from the test scripts.
- Structure scripts with minimal dependencies - Ensuring scripts executing unattended even on failures

D. General flow of data creation

A test case may or may not need the pre-requisite test data. Pre-requisite test data is nothing but the data that should already have been created or available before executing the automated test. The test data is also created by the application through automation. The general flow of data is as follows:

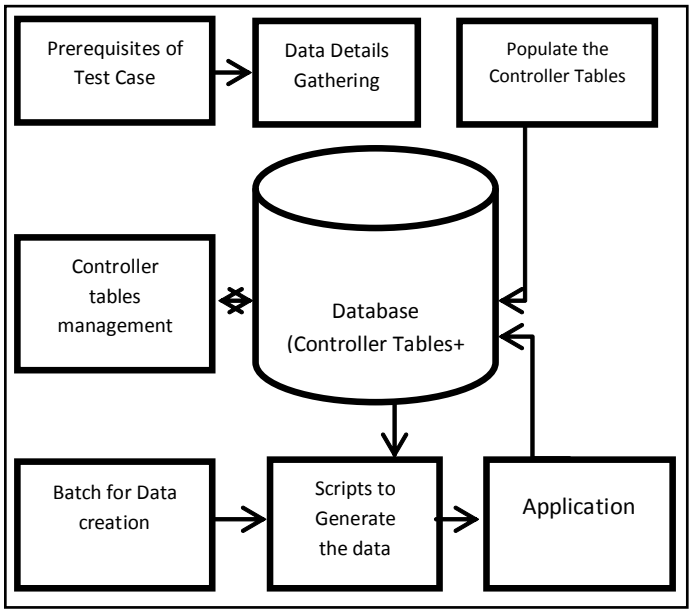

Fig. 2: Data Gathering and Test Data Creation Flow

E. Steps for Data Creation and Maintenance

While manual review of a test case from HP-QC the pre-requisites are analysed given in the details tab of the test case. If any pre-requisite data is needed, the details of the data to be created are gathered in a shared sheet by the test case developer. The Controller tables are the tables that are used for storing automation data and information. These tables are populated with the information that the application will use while creating the particular data. There are some indicators in the tables that will be used by data creating scripts to identify which data is to be created and which is not. A batch is prepared for creating the data. This batch is nothing but a VB script which calls other multiple scripts according to data creation needs. It is a VB script because we are using HP-QTP as automation tool for data creation of the AUT. Scripts in the batch runs one by one and make the connection with the database then fetch the information about the what kind of data is to be created, in what amount from the controller tables[5]. Then the data is created by the application through automation. When the data is created, application tables are populated with the information about the data that is created. This information is used during execution of the test cases for which the test data has been created. Once the test data is ready, the test developer can run the automated test so that it will not fail because of data validation issue or data unavailability issue [6].

III. The Proposed System

A. Framework Design

To start the testing with QTP, there is an important concept to understand i.e. framework. Frameworks basically define the way to handle the different operations in different methods. The test can be created in two ways in QTP i.e. by recording and by scripting. [6] We should also know that how we can improve the quality

of test by using Framework concept. QTP supports five kinds of framework i.e. Linear, Modular, Keyword driven and Data driven and Hybrid Framework.

1. Linear Framework

Linear Framework is very easy and catchy. Basically Linear Framework deals with individual script which is recorded under one Action and running individual. So this is very simple and easy way to create the test by navigating through the application.

2. Modular Framework

Sometime when we test some business processes in one application, we may feel that some operations we repeat in all Linear Framework Scripts. To handle this situation we go with Modularity which solves this kind of problem by dividing the single test in multiple parts or modules.

For implementing modularity, we basically divide the test in different parts so that we can form the functions for reusing. That solves many problems of repetition of some actions and gives the facility to make the script reusable. So in one sentence, Division of Linear Framework script in different parts called Modular Framework.

3. Keyword driven Framework

Tester should know the basic of programming because Keyword driven Framework deals with functions. In programming, function is an important part of programming as they allow creating chunks of code that performs a specific task. Basically we create the functions forming the function library and calling these functions as keywords in the tests therefore this becomes keyword driven framework with modularity. This is very important and useful framework to deal and handle many critical tasks in testing through QTP.

For example: Consider that we have to automate a flow where we would need to do the following things in a Mailing Application.

- Login to Application.
- Count the number of unread emails in Inbox.
- Logout from Application.

If we look at the above test case from pure Modular Framework point of view, we will be able to easily identify that we would need to write 3 different functions for the above test case. These 3 functions will be used to Login to Application, count unread emails and logout.

Once we have identified the functions, the next step is to identify some keywords and then associate the functions with these keywords. Below figure shows a pictorial representation of this concept.

Test Case Flow

>> Login to Application
>>Count unread mails
>>Logout from Application

Identify keywords for each flow
Keywords can be anything with a meaningful name as follows

>>App_Login
>>Count_Unread_Mails
>>App_Logout

TestCase flow according to Keywords

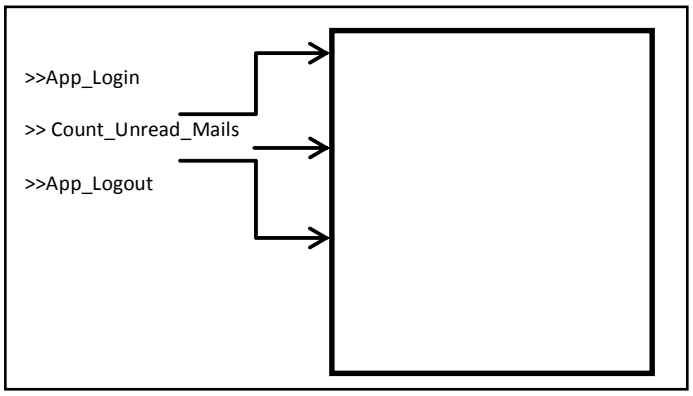

Fig. 3: Mapping between keywords and functions

Each keyword is associated with a separate function as shown in the figure.

From the above image we can see that basically a Keyword Driven Framework is nothing but a collection of keywords & functions (or actions or operations) and the association between these two entities.

4. Data driven Framework

Data Driven Testing Framework is a framework which is driven by test data, which means that test data is the important factor here. The basic expectation with this kind of test automation framework is scripts should be built in such a way that it should work for different sets of data without any changes to the test script.

Test cases that are executed multiples times—once for each input of a given set of data use data driven approach. In Linear Framework because the data is hard-coded within script, we cannot use the same code with multiple data values. We overcome this shortcoming by using Data Driven approach.

Consider a scenario where we have a large number of user credentials for an application. We need to login to the application with each of these credentials to find out which of the user credentials are working properly and which are not. In order to complete this task, we can create a script in QTP which would read each of the user credentials as variables and try to login to the application and at the end would report the results.

5. Hybrid Framework

In this approach, we mix data driven approach with modular approach or with keyword driven approach. Scope of Hybrid framework is very high as we mix different approaches. This approach is flexible for performing different tasks.

B. MoKey Framework

MoKey is a modular keyword-driven automation framework. The current version of MoKey is implemented using QTP and VBScript.MoKey combines industry best practice concepts like modular programming and keyword-driven testing into a single hybrid framework MoKey reduces and simplifies automation test and code maintenance. These tests are readable and easy to debug the tests. It reduces the training requirements for new users.

C. MoKey Structure

The user interface to MoKey exercises the keyword-driven testing concept. The code supporting the keywords is implemented using

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

the modular programming concept.

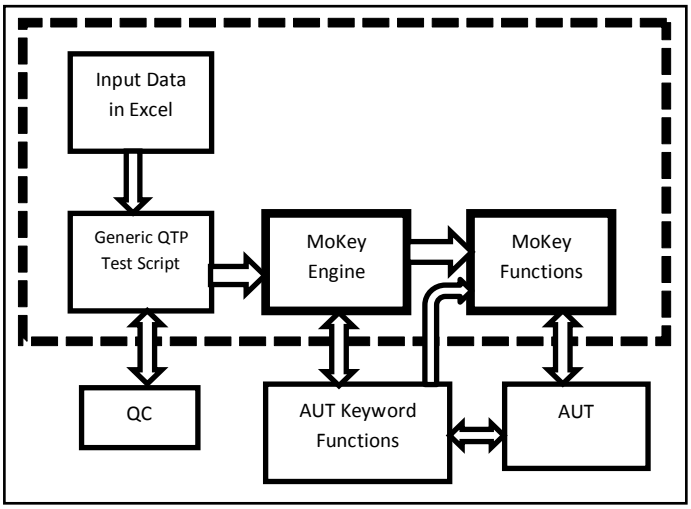

Fig. 4: MoKey Structure

QTP Automated test:

Browser("Welcome:Mercury Tours").
Page("Welcome:MercuryTours").WebEdit("userName").Set "keith"
Browser("Welcome: Mercury Tours").Page("Welcome: MercuryTours").WebEdit("password").SetSecure "4678374e3e3faccb50be844f56637acb93d4fa07"
Browser("Welcome: Mercury Tours").Page("Welcome: Mercury Tours").Image("Sign-In").Click 38,9
MOKEY Test Script:
Call startTestScript ("D:\Automation\Demo\MercuryTours\Data\data1.xls")

MOKEY Data File:

All MoKey tests consist of a QTP test script and a data file as shown above. The 1-line QTP script references the data file name.If we automate that manual test case with MOKEY, it will result in an automated test script and a supporting input/validation data file. The test script will be one simple line that points to the supporting data file. The data file comprises both the control and test data and is stored in an Excel file. The data is represented as a triplet of keyword, object name, input/validation data. The latter two are considered supporting parameters for the keyword.

Keyword - represents a typical action or validation that a tester performs during a test step and can also represent business processes or tasks. Keywords are used instead of the automation tool's functions/methods.
Object - represents the object name that the keyword operates on.
Value - represents the input or validation data for the object. Action - represents the default action performed on an object validate - represents the default validation of an object's attribute
Control Data: The data or part of the automated test that performs actions on AUT is called as Control data.
Test Data: The data that is provided as input to the automated test or perquisite is called as Test Data.

Table 1: Test case datasheet written to be executed using MoKey framework

Keyword	Object Name	Input/ Validation Data	Comments
StartTest			Start Web Application
Action	Username	John	Enters value into user name field
	Password	Mypwd	Enters value into password field
	Sign-In		Clicks on Sign-In
Validate	BrowserTitle	Find a Flight	Validates whether correct screen Appears
EndTest			Ends test

D. Components of Framework

1. Function Library

Function libraries usually play a very important role. All the necessary intelligence is built in the function library so that it can read the excelsheets and call the different functions from the function libraries itself based on the Keywords. And so provides modularity.

2. Object Repository

Based on the design of Framework, we can use an object repository. An object repository is a very essential entity in any UI automation tool. A repository allows a tester to store all the objects that will be used in the scripts in one or more centralized locations rather than letting them be scattered all over the test scripts. An Object repository is a file that maintains a logical representation of each application object that is referenced in an automated script. The purpose of the Object repository is to allow for the separation of application object property data known as the Physical Description, from the automated script via a parameter known as the Logical Name.

E. Key Benefits of Framework

Standard process in Production: Due to use of Test automation framework, a single standard is established across the organization. This helps the organization as the standard processes are followed as compared to pre-empted ad-hoc processes, which yield no results.

1. Free from Dependencies

Complete coding and component usage standards are defined in production. Independency from the individual coding standards and the utilities/components created. Complete documentation helps the organization in inducting the new members with minimal effort.

2. Complete Coverage

Requirements are collected from an overall application's perspective. This overall coverage minimizes the testing effort

during the later stages of the releases, for the entire product suite across the organization.

Future Enhancements Support: Automation team need not worry about testing future enhancements. Only minimal changes and the validations related to the enhancements need to be added to the existing base framework and that too with minimal effort.

Cost Estimation: The cost includes,

- Acquisition cost - In the procurement process of the tool, following cost needs to be considered: Tool Cost, Cost based on number of licenses, based on our requirements and Version Upgrade Cost.
- Training - Training cost incurred for training test engineers, business users, developers and creating supportive training documentations must be taken into consideration.
- Environment - Cost involved in setting up the system environment (Hardware and Software) must be taken into consideration.
- Development - Development cost can be calculated based on the components designed in the framework development.
- Maintenance - Each tool has its own maintenance requirements.

IV. Conclusion

Test automation implemented during this project accelerated the regression testing cycle and promoted software quality. Automating tests and other repetitive tasks reduced time required in regression testing and expanded the quality of testing. Test coverage is increased by extending automation to parts of the application that may not have been thoroughly tested in prior releases. The use of Modular Keyword Framework and hybrid automation approach further increased testing efficiency.

The Automation approach used for pre-requisite test data creation has reduced time consumed in manual creation of test data. Pre-batch developed for checking availability of test data before running the test avoided tiresome work in manually checking test data availability and thus drastically reduced the time consumed. Test Automation required less manpower to perform tedious processes and therefore cut man hours and costs.

V. Acknowledgement

This research was supported by Oracle India Private Limited,Bangalore during my internship program (August 2013-December 2013) in Unified Automation testing team- RGBU.

References

[1] http://www.automatedtestinginstitute.
[2] http://www.automationrepository.com.
[3] http://www.qtpselenium.com/
[4] http://www.softwaretestingmentor.com/automation/
[5] http://www.oracle.com/technetwork/tutorials/index.html
[6] Mark Fewster& Dorothy Graham (1999). Software Test Automation. ACM Press/Addison-Wesley.

A.Divya received the B.E degree in computer science engineering from Velammal College of Engineering and Technology, Madurai, India in 2012.She did her Internship as a part of her Master's degree in Oracle India Private Limited, Bangalore in 2013.She is currently a student pursuing Master of engineering in Mepco Schlenk engineering college, Sivakasi, India. Her research interests include Network Security, Wireless Security and Networking.