# A FRAMEWORK FOR ENHANCING PERFORMANCE AND MINIMIZING AUTOMATION TESTING COST AND TIME USING EFFICIENT METHODS

*Thesis submitted in partial fulfillment of the requirements for the award of degree of*

**Master of Engineering**

in

**Computer Science and Engineering**


*Submitted By*

**Tavleen Kaur**

**(Roll No. 801332030)**

Under the supervision of:

**Dr. Shivani Goel**

Assistant Professor



COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004


**July 2015**

## CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled, "***A Framework for Enhancing Performance and Minimizing Automation Testing Cost and Time by using Efficient Methods***", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Computer Science and Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Shivani Goel* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Tavleen Kaur

Signature:

(Tavleen Kaur)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Shivani Goel

(Dr. Shivani Goel)

Assistant

Professor CSED

**Countersigned by**

**(Dr. Deepak Garg)**

Head

Computer Science and Engineering Department

Thapar University

Patiala

**(Dr. S. S. Bhatia)**

Dean (Academic Affairs)

Thapar University

Patiala

## ACKNOWLEDGEMENT

The satisfaction that accompanies the completion of any task would be incomplete without naming the people who made it possible with their constant guidance and encouragement.

I am extremely thankful to my respectful Guide Dr. Shivani Goel, Asst. Professor, CSED, Thapar University, Patiala for her valuable advice, motivation, guidance, encouragement, moral support, sincere efforts and attitude with which she solved my queries and provided delightful ambiance for learning, exploring and making this thesis possible. It has been a great experience and pleasure to work under her sanctuary.

I would also like to deeply express my gratitude towards Dr. Deepak Garg, HOD, CSED, Thapar University, Patiala, who has always been a source of motivation and hope.

Above all, I would like to bow and salute God, who is ever present with all and showers His blessings.

Tavleen Kaur

Software Testing is an important part of software development process. The aim of software testing is to find bugs and errors in the software. It also ensures that actual results are matching with the expected results and the requirements specified by the customer are being satisfied by the software developed. Software testing also tests the software for factors like efficiency, performance, response time, compatibility, security, reliability etc., and hence, ensures the quality of the software.

In recent times, the technology has advanced to a larger extent, which in turn, has led the industries to require much complex and riskier software/applications, which also requires improvement in quality of software. So for this reason, to find the maximum bugs in software in lesser time, many organizations have adopted the technique of Test Automation.

There is always some cost involved in everything. Testing and maintenance costs are approx. more than 50% of total cost of software development. If chosen the wrong methods and process for test automation, it can lead to 70-80% of total cost of software development, which could be out of the budget and efforts and time will also be wasted. So, the tester must ensure that the methodologies, process and framework used for the automation are the best and efficient enough so as to reduce the costs and increase the performance.

The work that has been carried out in this thesis is focused on the implementation of some of the efficient methods and the framework, which can help in achieving the above stated purpose to a great extent. Following the correct process in the early and initial phase of the testing can help in cutting down the cost, efforts and time and increasing the performance of test automation.

## TABLE OF CONTENTS

## LIST OF FIGURES

## LIST OF TABLES

# Chapter 1

# Introduction

## 1.1 Introduction to Automation Testing

In recent times, the technology has advanced to a larger extent, which in turn, has led the industries to require much complex and riskier applications, which also require improvement in quality of software. So for this reason, to find maximum bugs in software in lesser time, many organizations have adopted the technique of Test Automation. So, Automation Testing has now become very popular in current time and almost all the software industries are following it.

Automation Testing is defined as the process of testing software, where all the steps and functions, such as, initialization, performing some steps, execution, validations, analysis, results verification etc. are performed by automation tools automatically with the help of test scripts. [1]

### 1.1.1 Manual Testing vs. Automation Testing

We can perform testing manually as well as automatically. But research puts more emphasis on automation testing which helps to test complex functionalities with less effort. [5] Table 1.1 summarizes differences between manual and automation testing.

**Table 1: Comparative Study of Manual and Automation Testing [5]**

| **Factors** | **Manual Testing** | **Automation Testing** |
|---|---|---|
| **Time** | It is time consuming & tedious, because testers have to do each and every step manually | It is quite faster than that of manual testing, since tests are performed by the automation tool |
| **Resources** | It requires more human resources to run tests | It requires less human resources to run tests |
| **Efficiency** | Less reliable: it is prone to human errors | It is more reliable and produces accurate results |

| | | |
|---|---|---|
| **Execution Frequency** | Difficult to run the test cases for a large number of time | Easy to run the test cases for any number of time |
| **Effort** | Effort is same every time of testing | Effort is large initially but then it is reduced very much |
| **Cost** | It requires large investment in terms of human resources | It just requires initial setup. After it is done it can be re-used any number of times |
| **Training** | Training for developing scripts is not required | Testers need to be trained to use automation tool to develop scripts |

### 1.1.2 Benefits Provided by Test Automation

The main purpose of automation testing is to automate the manual testing process which is repetitive, difficult and laborious. Main use of automation tools is to automate regression testing(where only new part changed is testing again). For this, there should be a database that contains the test cases that are repeatable. Thus we can run this test suite every time whenever there are changes in the application to make sure that any unintended consequences are not produced by the changes made. Hence, we cannot say that automation testing is a replacement for manual testing. Instead, it is the continuation of manual testing to get higher speed as well as accuracy in entire testing process. [3]

The main reasons and benefits provided by test automation, which has attracted a lot of software industries towards it and made automation testing process interesting and beneficial for an application are:

- Manual Mistakes: Test Automation reduces the possibility of human errors introduced by the user which may occur because of mistakes or misunderstanding of requirements, hence gives better reliability. This is the common problem because testers generally have to undergo peer pressure to complete set of executions in less span of time, which may lead to confusion because of huge amount of test cases and their complexity to execute etc.

- Parallel Execution: In comparison to manual testing, parallel execution is possible with test automation as a number of tests can run simultaneously in automation testing. Hence, less number of resources are required wherein the work will be done faster.

- No Interruption: Automated test scripts can be executed all-night long without any human intervention, which in turn saves times.

- Easy Result Analysis: Just running the test cases is not the purpose of testing, but testers have to analyze the results thoroughly and may be sometimes, they have to forward their analysis to the developers in order to fix the bugs. Test Automation could be followed by automated analysis, where the automation tool can show passed, failed, blocked test cases along with graphs, snapshots of errors, which makes it very easy to report the results. [3]

### 1.1.3 Test Automation Principles

As the software complexity is increasing, automation testing is becoming more popular. Because of current increasing demand for efficient and bug-free products, it is more crucial to test each scenario of the software.

Survey on testing shows that advantages of test automation are basically related to re-usability of tests, repeatability and saving in time and effort in execution of tests. However, the limitations are initially high investment set-up is to be done, then best tool has to be selected and training for that tool is required.

Automated software testing means to automate the manual process of software testing. The basic terms which are used in automation testing domain are:

- Test Automation: It is developing the software for testing of the software product.
- Test case: Test case consists of a set of steps which are to be performed to check the software is functioning and to find the errors
- Test data: The seed data is to be used in test cases to test all the possible scenarios

- Test suite: A set of test cases is known as test suite [4]

### 1.1.4 Automation Test Process

There are many phases to be followed for the implementation of any automated tool. Each phase defines a particular activity and each of this gives a definite outcome.

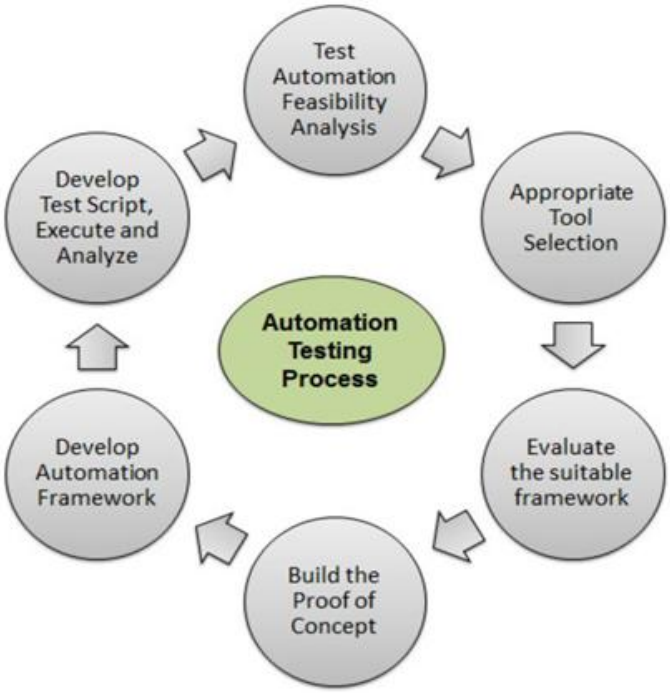



**Figure 1.1: Automation Test Process [5]**

- Test Automation Feasibility Analysis: This is the first step, where it is checked that whether the application can be automated or not, because all the applications cannot be automated

- Appropriate Tool Selection: Selecting the best and appropriate tool is the next step. This selection is basically dependent on the features, usage and requirements of application which we need to automate

- Evaluate the suitable framework: After selecting the tool, now a suitable framework is needed to be selected

- Build the Proof of Concept: Then Proof of Concept (POC) is developed with complete scenario to see whether the selected tool supports the automation of application

- Develop Automation Framework: After POC is developed, automation framework is developed, which should be developed after analyzing the features and technology used by the application

- Develop Test Script, Execute and Analyze: Test script is developed, executed and then results are analyzed and if there are defects, these are logged [5]

## 1.2 Factors Affecting Automation Testing Cost and Time

The idea that instead of doing manually, computer will run the tests has lead to organization follow this concept without the actual and proper understanding of the other factors involved. Automation Testing can bring many benefits: it can do more work in less time and less resources, reusability, makes it easy to find bugs quickly in early stages and the results are more reliable. But on the other side, it requires one time large investment in terms of time, expertise and resources to make the process of automation testing successful overall. So, it is always best to keep in mind all these factors before starting Test Automation.

"Reducing Testing Cost" is the buzzword nowadays in software industry. The cost of testing can be minimized to an extent using mainly two ways- through test optimization and by increasing the productivity. Test optimization means using efficient test methodologies, reusing components and reducing redundancy and by using optimized test data and test suite. Whereas, test productivity means to increase the productivity of the process of testing. [6]

This is the goal of each testing team in each software industry:

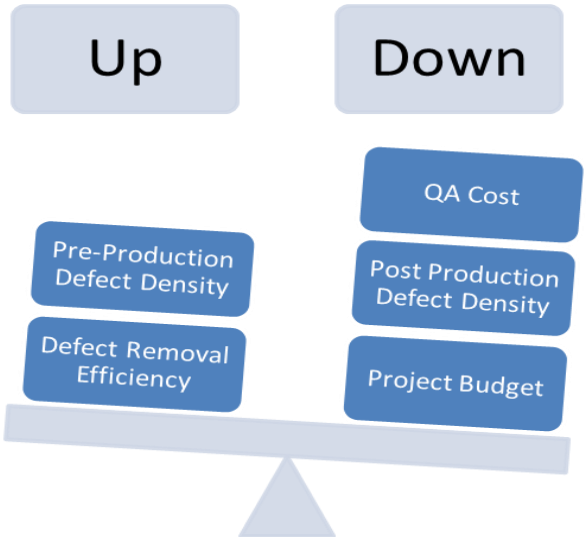



**Fig. 1.2 Testing Team Goal [6]**

There are various factors which affect Automation Testing Cost and Time. These are:

- Choice of Automation Tool
- Design of Framework
- Test Case Selection in Test Suite
- Testing most Bug Prone Areas of Application
- Results Reporting, etc.

So, the basic problem covered in this document comprises of various efficient methods to be used for all above factors so that Automation Testing becomes less costlier area in terms of money, resources and time and at the same time, provides better results.

## 1.3 Thesis Outline

In this thesis report, **Chapter 2** covers the state of art in automation testing, where various studies from literature are studied for my problem statement. It covers the current ideas, methods and techniques used for automation testing in the organization.

**Chapter 3** covers the gap analysis between current work and the desired work to solve the problem and the problem statement in brief is discussed here along with the reasons why this problem has been chosen.

**Chapter 4** covers the design and implementation part to solve the problem under consideration, where methods are suggested to solve the problem.

**Chapter 5** describes the implementation results of the methods.

**Chapter 6** concludes the document with analysis on results derived. Also it covers the future scope of the problem and what research in this area can be carried ahead. At the end of the document, References and Research Paper Acceptance and Communication details are mentioned.

## Chapter Summary

In this chapter, a basic introduction to Automation testing is given. Then a comparative study is presented between manual testing and automation testing. Based on this the benefits are listed out which we get from automation testing over manual testing. Then the process followed by test automation is described. This thesis is about improving performance of automation testing. So, various factors are listed out which directly or indirectly affect automation testing. These factors will be discussed in detail in coming chapter.

# Chapter 2

# Literature Review

## 2.1 Record and Playback Testing

Test Automation means to automate the manual process, which uses a well-defined testing process. This process should be having:

- Detailed Test Scripts: The test scripts contain test case description, preconditions, initializations, steps to perform particular functionality, validations and expected results. These test scripts should be based on functional specifications
- A Test Environment, which is dedicated

In the method of Record and Playback, tester performs the steps of each test case manually and records it with the help of automation tool. This generated script is run again whenever required and the results are validated for the expected results. This method does not require any automation knowledge and special expertise. This method is best suitable in which:

- Scripts are to be created and to be used for a lesser span of time. Long time is required for the increased efforts for scripts maintenance
- The resources are not expert in automation, but test automation is compulsory to be used for the project
- If the test scripts will be executed for just few execution cycles temporarily and will be discarded later on [7]

## 2.2 2 Improving Test Automation Cost-Effectiveness

Initially, due to several advantages of automation testing, software industries want to automate all of their test cases so that they get executed faster in less amount of time.

But, it should be always kept in mind that the initial investment needed to automate the test cases is initially very high, and sometimes the proportion of cost-benefit of test automation could also be low in some situations.

So, before starting the test automation process, one must do pre-automation analysis. First, all the test cases could not be possible or advantageous to be automated. So, it is necessary to check that which test cases are technically possible to automate and which will be advantageous too. Secondly, it is important to check the test cases analyzed in first step are viable to be automated or not. A test is said to be viable to be automated, if it gives the same result through automation as it gives manually.

So, proper analysis and making appropriate decisions can prove critical for the successful test automation process. [8]

## 2.3 When and How to do Test Automation

Automation testing is considered to be a long-term solution for reducing software testing cost and time but getting better quality product. But this goal can be achieved only when some best methodologies are followed before starting automation and during automation.

### 2.3.1 When to go for Automation

Even before actually starting the process of automation, one has to spend a lot of time and efforts in order to ensure that things mentioned below are taken care of:

- The application should be stable in terms of its functionality. There is no use of spending time and efforts on an application, which keep on changing functionality-wise.
- The application can be on any interface such as command-line, graphical-user based or application-programming interface. One has to identify in the beginning which type of interface the application is having, because automating different interfaces require different efforts and techniques. So, one has to plan the testing process accordingly.

- Scope of testing should be decided in terms of coverage of application. It could be either functional testing or load testing or regression testing or unit testing.

### 2.3.2 How to do Automation

When the above points are taken care of, right direction has been verified, one can proceed with automation. Then following points should be taken care of:

- A mini-development cycle must be followed in automation testing. So, one must prepare proper coding guidelines and standards to code the scripts and follow these
- The automation suite should be organized in such a way that it should be flexible and more test cases can be added to it at a later stage
- Our test scripts should generate proper logs in case the test scripts fail at some time describing the reason of failure
- All the test cases should be independent ideally and should not depend on the successful completion of some other test case(s)
- After test case execution, automation test suite must bring the application back to the stage where it was before the application execution started [9]

## 2.4 Test Automation Framework Development Challenges

To develop a framework for test automation is a challenging and multi-stage process. Each stage leads to challenges faced by the developer. The challenges which need to be addressed can be following:

- Clear Vision: One must have the clear vision in terms of what needs to be achieved from this automation. Vision should be defined and well documented. To have the vision clear, these things must be identified:
    - Testing Model to be used
    - Types of testing to be performed
    - Prioritizing the testing activities based on requirements (for example, performing smoke testing and then regression testing)

    - Which areas of application need to be covered in automation
    - Challenges due to functionality of the application

- Tool Identification : It involves several other factors to be considered as following:
    - A tool evaluation checklist should be created based on testing type, cost of tool, extensibility, performance, training etc. factors
    - Testing requirements: functional, regression, smoke etc
    - Multiple tools may be needed to perform different types of testing on different products.
- Framework Design: It requires to identify the requirements from several areas:
    - Identify reusable components of application
    - Type of input data
    - Communication between application components [10]

## 2.5 Impact of Ineffective, Inefficient and Unsecure Test Data

The best way to ensure that applications will work well in real-world situations is to use the same kind of test data that will be used in a live application. So, the data must be taken from the production system to validate and test the functionality, performance and logic of the application. But there are not any standards to capture the right data for testing the application. So, normally organizations do testing with large amount of data, wrong data or unsecure data, which bring critical consequences for both business and IT. [11]

**Table: 2.1 Impact of non-optimized data on IT and Business [11]**

| **Issue** | **IT Impact** | **Business Impact** |
|---|---|---|
| Large Amount of Test Data | • Excess memory required for storage and processing<br>• Costly to refresh data | • Unpredictable and wasted spending on data<br>• Less revenue<br>• Poor application quality |

| | | |
|---|---|---|
| | | is a risk |
| Poor Coverage of Test Data | • Application logic not checked completely<br>• Wasted execution cycles<br>• Poor productivity from staff | • Customer satisfaction is reduced<br>• Application's live performance can be affected and hence organization's reputation will also get affected |
| Unsecure Test Data | • Data security risks can affect other applications also<br>• Third parties can misuse sensitive data<br>• Data integrity issues | • Fines and penalties to organization due to data breach<br>• Lost customers<br>• Reduced reputation |

## Chapter Summary

This chapter summarizes the various literatures studied and reviewed which are related to the topic under research. Record and playback testing has been the traditional technique of testing for many of the testers. So, a study on record and playback testing is reviewed and discussed in this chapter. Then the important points like when and how to go for test automation have been discussed. This thesis intends to implement a test development framework. For this purpose, various challenges that are faced in framework development are discussed here. Then at the last how can the wrong test data degrade the business and IT, these impacts have been listed out.

# Chapter 3

# Problem Statement

## 3.1 Research Problem Statement

**"*A Framework for Enhancing Performance and Minimizing Automation Testing Cost and Time by using Efficient Methods*"**

This thesis is intended to use the framework and other efficient methods in order to overcome the high costs and time taken for test automation and enhancing the performance of test automation as well.

## 3.2 Problem Description

"Reducing Automation Testing Cost" is the buzzword in the software industries these days. But Performance should not be compromised; it should be enhanced keeping cost and time at its least. [12]

There are various factors which effect automation testing cost, time and performance directly or indirectly:

### 3.2.1 Choice of Automation Testing Tool

The performance of test automation process largely depends on the selection of the right test automation tool

#### 3.2.1.1 Gap Analysis - Choice of Automation Testing Tool

There are various testing tools available in market for testing the software or applications. Some of the tools are licensed and some are open-source. Each tool has its own characteristics and features. Depending on the requirements of the software, the tester has to choose right testing tool for his software. [13]

There are some basic criteria on which we can judge the performance and suitability of testing tool. This thesis lists down those criteria and compares two automation testing tools- HP QTP and OATS based on these criteria. These criteria list can be used to compare the performance and suitability of any other automated tools.

### 3.2.2 Design of Framework

A framework defines the flow of work, how the things will be carried out to perform a particular thing. The framework becomes a standard, which is followed by every other person involved in that work.

#### 3.2.2.1 Gap Analysis - Design of Framework

The proper and well-designed framework should lead to less maintenance, more reusability, better performance and more efficiency. When the automation is started, there is no framework initially, tester has to decide and define what should be the flow of data, and functions etc. so that it requires less effort and time for automation, but give more performance at the same time.

Currently, Record and Playback is used for writing the scripts, which is not at all efficient. It leads to redundant code, and hence, more maintenance. So, the idea is to use a framework composed of reusable components with efficient flow of data across the framework. [14]

##### 3.2.2.1.1 Drawbacks of Record/Playback

- Recorded scripts are very difficult to maintain. Recorded script is very long which contains actions on the objects present on user interface, which ultimately are tough to decipher
- Synchronization is an important problem with test automation. It means a window may take time to appear when playing back the script. So, recorded script can lead to synchronization issues. If window will not appear on time when playing back the script, the script may perform wrong steps on opened window or it may fail

- Recorded script contains hard-coded data values in the script, which is not a good practice
- An ideal test script should handle all those scenarios what may happen; it should not only cover what was happened while recording the script. Recorded script lacks in this point. The main objective of test automation is to find bugs which can occur at any time in the system. Using record/playback, a script will just handle the errors that were encountered while recording the script, while the test script should handle all unexpected errors also through the programming code.

When a test case is automated using recording, it generates script lines. Thus, the relationship between two is no. of lines of the script is directly proportional to the no. of test cases automated:

$$\text{LinesOfAutomationCode} \; \alpha \; \text{TestCasesAutomated}$$

More the test cases recorded, the more is the code to maintain; sometimes it reaches to a point here maintenance cost overcomes the budget. [15] Following figure shows this relationship graphically:

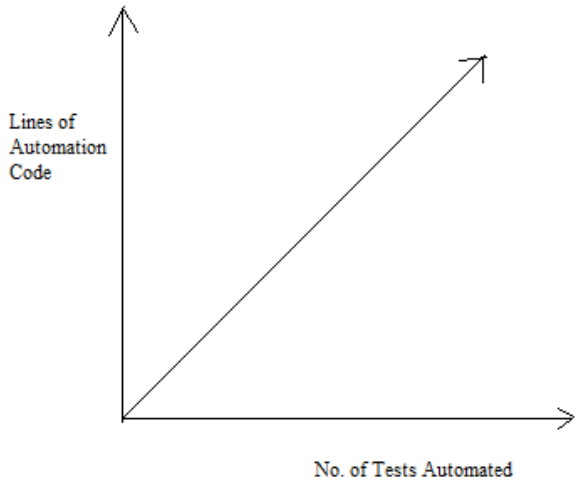



**Fig. 3.1: Graph of relationship between no. of tests automated and lines of automation code [15]**

It also shows that the code cannot be re-used; since all the tests have their own dedicated scripts.

### 3.2.3 Test Case Selection in Test Suite

A test suite contains multiple test cases covering different functionalities of the AUT. The selection of the test cases should be done properly to be included in the test suite or not [16].

#### 3.2.3.1 Gap Analysis - Test Case Selection in Test Suite

There can be multiple scenarios according to the functionality of the application to be tested. But it is not necessary that all the scenarios can be automated. And also some test cases are not beneficial to automate.

There are some factors that need to be checked against each of the test case to decide whether it should be automated or not. Otherwise, if wrong test cases are chosen to be included in the test suite, then it may lead to wastage of resources as well as effort and time[16].

### 3.2.4 Testing most Bug Prone Areas of Application

There are some areas in an application which are most bug-prone for e.g., transactions in accounting system. These areas must be tested more frequently than that of other non-bug prone areas to assure better testing of the system. [17]

#### 3.2.4.1 Gap Analysis - Testing most Bug Prone Areas of Application

Whenever a version of product is released in the market, user uses it and logs any bugs found in the system. Testing team must focus on the areas under which more bugs are logged. If testing is done for the whole system covering all scenarios again and again, it will just result in the wastage of resources, efforts and time. So, it is best that the areas which are most bug-prone should be segregated and should be tested more frequently so that the bugs are identified and removed fast from those areas. [17]

### 3.2.5 Results Reporting

Before the release of the version of an application, testers have to perform exhaustive testing for various code-drops for multiple times. A batch covering multiple test cases is run over-night to cover maximum possible scenarios and if parallel execution is possible, multiple batches are run simultaneously.

#### 3.2.5.1 Gap Analysis - Results Reporting

There are multiple teams and members who are associated with the testing of the product or application. The reports generated after running a test batch have to be communicated to all interested and concerned teams and members. It is very tedious and time-consuming to manually send the reports to each person. So, the idea is to have automated mail service which at the moment, when batch will finish the execution, will send the reports generated to all registered people. It will help in saving the time and effort.

## Chapter Summary

In this chapter, the main problem statement for which research work is being carried out is described along with the gap analysis for problem and sub-problems i.e. why this topic has been chosen to be researched.

# Chapter 4

# Problem Solution

## 4.1 Application under Test (AUT)

To conduct the research, the Oracle Retail application – RWMS (Retail Warehouse Management System) will be used. Here is the brief introduction to the application.

### 4.1.1 Retail Warehouse Management System

RWMS is a warehouse management system that provides various tools and facilities which are essential to control a modern DC i.e. Distribution Center(Fig 4.1). RWMS also optimizes and enhances the flow of goods and resources at the fulfillment center, Distribution Center or warehouse itself. RWMS functionality helps to enable fulfillment of requirements across multiple channels. The web architecture of RWMS provides functionality and visibility to various remote facilities and trading partners through the Internet. Decision support tools of RWMS help to plan using facility resources effectively and also monitor existing activities and flow of goods. Radio frequency (RF) terminals help to make task management and real-time inventory control possible. [18]

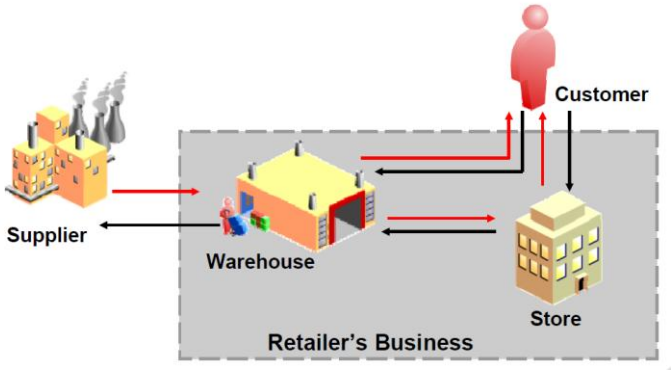



**Fig. 4.1: RWMS Overview [18]**

### 4.1.2 Different Tasks Managed by RWMS

The Oracle Retail Warehouse Management System has some primary processes/ tasks like:

- **Appointment Scheduling:** Required to receive the merchandise within the DC, there is designated time and location for trailers to drop off the goods within a DC
- **Receiving:** When the goods are physically brought into the DC
- **Distribution and wave Planning:** Where the merchandise is distributed within the warehouse and waves or groups of merchandise are planned, all the ensure effective movement of merchandise
- **Put away**: Is segregation of merchandise in a specific location for future movement or processing, as the need may be
- **Returns**: Refers to the return of inappropriate merchandise from the warehouse, sometimes the goods that are received may be damaged and cannot be processed in the warehouse, such goods need to be returned back to the suppliers.
- **Picking:** Refers to the physical picking of merchandise by pickers before merchandise is shipped out of the DC
- **Shipping:** Which is the activity of outward movement of merchandise to store and so on [18]

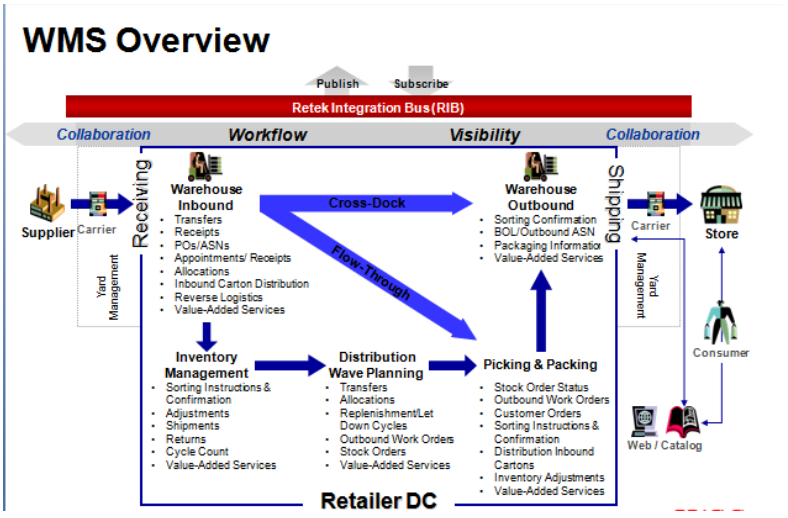



**Fig. 4.2: RWMS Process Flow [18]**

## 4.2 Choice of Test Automation Tool

Here, the objective is to choose the best suitable tool to test our application out of the tools available. For this purpose, a POC (Proof of Concept) will be developed. We have two automation testing tools available – HP QTP (Quick Test Professional) and OATS (Oracle Application Testing Suite). Comparative study of these automation tools will be done to choose the best suitable tool for the AUT based on several criteria such as efforts required to develop test scripts, playing back the generated scripts, cost, speed and the results generated by the tools etc. So, the major goal is to analyze the various features of these tools that assist to minimize the cost, time, resources required for the script maintenance and enhancing the performance. [19]

### 4.2.1 Research Methodology

The two available tools – QTP and OATS will be evaluated for their abilities in the form of a POC.

The different phases followed in this methodology are:

- Identifying the Application under Test (AUT)
- Define the steps for POC
- Define the criteria for comparison
- Select which tools to compare
- Run the sample test cases on both tools for comparison
- Compare results generated from the two
- Draw conclusion out of results [20]

### 4.2.2 Comparison of Version-specific Features

For the sake of comparison, the tools are installed and various features are tested. The exact features of the corresponding versions of the tools need to be aggregated. The table represents the version specific information:

**Table 4.1 : Features of OATS 12.4 and HP QTP 11 [21] [22]**

| Feature | OATS 12.4 | HP QTP 11 |
|---|---|---|
| Type of Tool | Regression, Functional and Load Testing Tool | Regression and Functional Testing Tool |
| Operating System Supported | Windows OS / Linux OS | Windows OS only |
| Application Type | Provides Inbuilt Database Supporting | No Inbuilt Database Support provided |
| Supported Browsers | Mozilla Firefox, Google Chrome, Internet Explorer | Mozilla Firefox up to version 17, Google Chrome up to version 24, Internet Explorer version 6 to 9, Netscape version 6.1 to 9 |
| Reporting | Inbuilt HTML and XML based reports | XML based reports and in HP Results Viewer |
| Back-End Tech | Java and Eclipse | .NET Technology, VC++ |
| License | Open – Source Tool | Licensed Tool – Seat/Concurrent License |
| Integrated Tools | Data Migration Tool, Create Supporting Package, Load Testing Agent Authentication, Database Configuration | Test Results Deletion Tool, Test Batch Runner, VB Scripting Engine, SQL Engine for DB Operation, MS Script Debugger |
| Ease of Installation | Easy to install | Requires patches to run for Web, Oracle, Java etc. after installation |

### 4.2.3 Automation Tools Comparison Norms

Use of automation tools enables developers and testers to efficiently robotize the testing process in SDLC. The objective of this research is to conduct a comparative study of automated tools such as the HP Quick Test Professional and the Oracle OpenScript based on parameters such as the efforts involved in generating test-scripts; ability of the tool to playback scripts; reporting of test-result; speed of execution; and cost of the tool.

### 4.2.4 Comparison Criteria

Automation Tools have the following features abased on which the comparison can be done:

- Utility
- Ease of learning
- User-friendliness
- Scripting features
- Recording Efficiency
- Ease of Debugging
- Playback Efficiency
- Test Results Reporting
- Data driven testing
- Re-usability and Maintenance
- Cost [20]

In order to make clear comparison between the tools, a 5-point scale will be used for each feature to calculate the average score 1(lowest) and 5(Highest).

#### 4.2.4.1 Utility

The Measure of usability of the tool mainly depends upon whether the client system requirements and configurations match with the functionality and technologies supported by the tool or not.

**Table 4.2: Utility Comparison [23]**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Supported Technologies | Web, Oracle PeopleSoft, Adobe Flex, Oracle EBS/Forms, Oracle Siebel, Oracle JD Edwards Enterprise One, | 1 | Web, WPF, Java, PeopleSoft, Oracle Siebel, Power Builder, SAP, Mainframe Terminal Emulators, | 4 |

| | | | | |
|---|---|---|---|---|
| | Oracle Fusion/ADF | | Silverlight, VisualAge SmartTalk, Windows Mobile, Web Services, Flex | |
| Supported Browsers | Mozilla Firefox, Google Chrome, Internet Explorer | 4 | Mozilla Firefox up to version 17, Google Chrome up to version 24, Internet Explorer version 6 to 9, Netscape version 6.1 to 9 | 5 |
| Supported OS | Windows OS, Linux OS | 4 | Windows Operating system | 2 |
| Testing Types Supported | Functional Testing, Regression Testing, Load Testing | 2 | Regression Testing, Desktop/Windows Application Testing, Web Application Testing, Database Testing, Mobile Application Testing | 5 |

#### 4.2.4.2 Ease of Learning

This is a very important issue, especially as most of the tools which have used in the past are currently in decline because of poor ease of learning feature. It is a very prominent factor in rating the tool's popularity. As a tool QTP and OpenScript both are easy to learn. OpenScript is tough for non-technical users; since it requires programming knowledge; whereas QTP can be learnt easily by non-technical programmers too.

**Table 4.3: Ease of Learning Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Ease of Learning | Comparatively tough to learn | 3 | Comparatively easy to learn | 4 |

#### 4.2.4.3 User – Friendliness

A large amount of the effort can be saved and best out of tool can be achieved, if the tool is user – friendly. Many times software products are successful in sale because of their ease of use. It not only saves the time, but also makes it efficient for the programmers to work on and hence increase their productivity. If the tool is difficult to learn, it directly a effects the comfort level of programmers and hence the efficiency and productivity are less.

**Table 4.4: User – Friendliness Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Access of Assets | Assets are to be attached to each script using Asset View | 4 | Object libraries can be associated only by adding the path to the script which is less efficient | 3 |
| Maintenance of Assets | All modifications in assets require only updating assets.xml file | 5 | If there are modifications in assets of a script, the code in the script must be modified | 2 |
| UI Features | Functions view, Action view, Keyword View | 4 | Step Grouping, Tree View, Assets View | 4 |

#### 4.2.4.4 Scripting Features

The features that ease the effort and time required for development are of utmost importance. The IDE support and popularity and features of programming language play a major role in the efficiency of the tool.

**Table 4.5: Scripting Features Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Scripting Language | Java based, hence Object-Oriented Support + | 5 | VB Script based, hence Object-based language, | 1 |

| | | | | |
|---|---|---|---|---|
| | Eclipse Editor Support (full featured IDE) | | which supports classes but doesn't support polymorphism and inheritance | |
| Scripting Mode | Script View only, which requires java programming skills | 3 | Keyword View - does not require programming skills,<br><br>Expert View - requires VBScript programming skills | 5 |

#### 4.2.4.5 Recording Features

All the features for recording are useful in learning how the tool works in default mode; advanced scripting is only possible after learning and discerning how tool interweaves the objects and code.

**Table 4.6: Recording Features Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Edit Code while Recording | Possible, but error while coding causes the recording to stop | 3 | Not possible | 0 |
| Recording Toolbar | Visible during recording, and provides many features like: choosing browser, stop, pause, play, inspect path, recording type | 4 | Not possible | 0 |
| Auto – Code | Yes in Java | 4 | Yes in VBScript | 4 |

| | | | | |
|---|---|---|---|---|
| Recording Modes | Only One Default Mode | 3 | Context Sensitive, Analog, Low level mode | 5 |

#### 4.2.4.6 Ease of Debugging

Debugging the script itself is a necessary process, and it is always beneficial if it is done quickly. The tools provide features to debug the issues. These features increase the productivity while debugging.

**Table 4.7: Ease of Debugging Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Checkpoints | Yes | 5 | Yes | 5 |
| Views | Console View, Properties View, Problems View, Details View | 4 | Watch, Output, Local Variables | 4 |
| Debug while Playback | Editing of code is enabled, but it won't be reflected in current run | 2 | Possible on occurrence of Error in the main script | 4 |

#### 4.2.4.7 Data Driven Testing

Nowadays data-driven testing (DDT) has become very prominent part of testing. Instead of creating multiple scripts to test different sets of seed-data, it is possible to cause scripts to access various sets of seed-data from external sources such as CSV files, excel sheets etc.

**Table 4.8: Data Driven Testing Features**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| External Data Access | Yes: Excel, Database, XML | 5 | Yes: Excel, Database, XML | 5 |
| Data Independency | Possible | 5 | Possible | 5 |

#### 4.2.4.8 Test Results Reporting

After execution of the test scripts, the results of execution provide information on basis of which it is possible to discern whether test scripts have passed or failed; and in-case they have failed, a proper analysis can be conducted hounding the underlying issue. So a detailed summary is expected from the tools.

**Table 4.9: Test Results Reporting Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Report Presentation | Detailed Summary in Results and Detailed View along with Screen-shots | 5 | Detailed Summary in HPQC along with Screen-shots | 5 |
| Checkpoint Information | Yes | 5 | Yes | 5 |
| Graphical Representation | Using Bar Representation | 4 | Using Pie Chart | 4 |

#### 4.2.4.9 Reusability & Maintenance

Reusability eliminates redundancy, and the lesser the redundancy, the greater is the ease of maintenance. Over a period of time, to accommodate the changes in the code, the impact on other code must be as less as possible.

**Table 4.10: Reusability and Maintenance Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Scripts | Since grouping of a test-script is done in steps, it becomes easier to find a particular step and modify it. | 4 | Step grouping is not done, hence it is little difficult to modify the big scripts | 2 |

| | | | | |
|---|---|---|---|---|
| Functional Libraries | Java supports polymorphism, hence in case a new method of doing the same task is required, it can be created in the same name using different signature. Maintenance of functions and extensibility is greater in OpenScript. JavaDoc comments makes it easier to use them | 5 | No polymorphism or JavaDoc comment equivalent feature exists. | 2 |
| Object Libraries | Easier mapping while renaming the objects | 5 | Manual change in code is required if name of object is changed in Object Repository, which is very inefficient | 3 |
| Other Assets | Any external file is also maintained in assets.xml | 4 | Any external file is added in the code itself. So changing the name requires modification in script. | 3 |

#### 4.2.4.10 Cost

QTP is licensed software which costs very high limiting; its usability is only in high-profile enterprises in small testing teams whereas OATS is an open source software.

**Table 4.11: Cost Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Cost | Open Source | 5 | Licensed | 0 |

#### 4.2.4.11 Playback Efficiency

There is no significant difference between playback features for both the tools. Time difference is seen when playing back the scripts in both tools which depend on time to load application initially, load inbuilt technologies such as java, response time of application and exceptions.

**Table 4.12: Playback Efficiency Comparison**

| Parameters | OATS | Pt. | QTP | Pt. |
|---|---|---|---|---|
| Synchronization | Yes through configuring think time | 5 | Yes through configuring wait time | 4 |
| Recovery Scenarios | Yes. Can be configured through programming. It is comparatively tough, but provides powerful control over error scenarios. | 4 | Yes. Can be configured through GUI | 3 |
| Efficiency over batch of 10 test cases | Very Good: 1112 seconds | 4 | Good: 1149 seconds | 3 |

## 4.3 Test Case Selection in Test Suite

The selection of the right test cases in the test suite can largely affect the performance of the test automation. All the functionalities and scenarios of an application are not possible to automate and moreover, to automate some of the scenarios might not prove beneficial. So, they just add to the efforts and time consumption of the tester and do not help in performance enhancement. [24]

### 4.3.1 Criteria for Test Automation

So, following points must be analyzed for each test case whether one should automate and put this test case in test suite or not.

#### 4.3.1.1 Execution Frequency

For each test case, it should be analyzed first that for how many times, this test case will be executed. In case, if it is going to be executed for one or two times only, then

it is useless to execute such test case. So, the scenarios whose execution frequency will be more are the best suitable candidates for the test suite.

#### 4.3.1.2 Generation of Reusable Code

When automating a test case, it is expected from it that it should contribute to the framework being followed. Even a test case that is very complex can prove good for automation, if it helps to enhance the efficiency of framework, i.e. the code of this test case will be reusable in other scenarios.

This criterion becomes more significant if the execution frequency of the test case is low. Because even if this test case will not be used solely frequently, but still its code will be reused at many places.

#### 4.3.1.3 Test Relevance

The fundamental goal of automation is to find the maximum number of bugs in the system. So, a test is said to be more relevant if it helps to find the maximum possible bugs in an application. So, the test cases that test critical and core functionalities and business process are the best candidates to be put in the test suite. For example, the test cases covering the banking transactions are more relevant test cases.

#### 4.3.1.4 Automation Effort

One must notice how much effort is being taken by the test case to make it automated. Unnecessarily more effort taken by the test cases can almost nullify the basic advantages of test automation.

But there could be some exceptions while analyzing this point as there are some reasons due to which it can prove beneficial to automate a test case even if it takes high efforts. Reasons can be: lifetime of test case, its execution frequency, reusability etc. So, these points should also be taken care of along with analyzing the effort required.

#### 4.3.1.5 Resources

The deployment of test cases also demands cost. So, one should have complete knowledge of the cost of deployment. Some test cases may need some high-performance hardware or new technology to be automatable. These factors may incur additional costs which could be out of budget. So, one should analyze what will be the

wise and right decision: spending on these requirements or leave the idea to automate and do it manually only.

#### 4.3.1.6 Manual Complexity

There exist many complex applications in the market, which require special training for the manual testers to test it manually. So, the test cases covering such complex functionalities are the best candidates to be automated; since not everyone will be able to perform manually every time without training, whereas if test case is automated, it can be run anytime by any one, without having deep knowledge of the functionality.

#### 4.3.1.7 Porting

If the environment gets changed, where the test cases are developed and being executed; it can create problems and testers may have to do a lot of re-work to make them work again. So, this is the biggest challenge to choose such test cases in the test suite which will be functional even if the product interface or environment will change.

Normally, the critical business processes and functionalities of the application remain the same throughout its multiple versions and releases. To include these test cases in test suite for automation can prove very beneficial.

But to choose the test cases which are possible to be generic to be ported to as many environments as possible is the best practice to be adopted.

#### 4.3.1.8 Execution Effort

Execution effort while test is being executed should also be taken into consideration. If the automated test runs faster than that of manual test, then it is good thing, but it may not happen every time. Sometimes, the automation execution takes more time than the manual one. But again with this point, you may need to consider other points also. If you think that execution effort can be compromised over the other benefits of the automated test, then one should go for automation, otherwise, it is advisable not to include such test cases in test suite.[24]

### 4.3.2 Problem Solution

The concept of Data Mining will be used for deciding which test cases to automate and which to not. The solution is to design the model based on the historical data, on the basis of this model, prediction will be done should or should not to automate the test case given the criteria's. The steps are as follows:

#### 4.3.2.1 Historical Data

We have the historical data, which contains the information like based on the factors discussed above whether the test case was automated or not for each of the test case.

#### 4.3.2.2 Model Design

Using this historical data, a model will be designed using which the predictions whether to automate the test case or not will be done. To design the model, following steps are followed:

- First sampling of data is done to check the accuracy of data. For this purpose, 90% of data is taken as the trained data and other 10% is taken as the test data
- Since we have the categorical data (in form of 0's and 1's), we will use logistic regression here. In case of continuous data, linear regression in used
- Logistic regression is used to predict the probability of the output, which can have only 2 values. This prediction is made based on given several factors
- So, here 10% test data is checked over the 90% trained data and the predictions are noted. Wherein we already have the actual outputs for these 10% test data. Moreover, it is not necessary is that all the factors are necessary to be considered while building the model. Some of the factors may not be that useful and can lead to less accuracy. So, one has to find the model consisting only important factors that lead to high accuracy
- These actual and predicted outputs are matched to check the accuracy of data. If it is, say, more than 75%, then data is considered to be accurate else whole process of sampling is done again with different factors till the point we don't get the desired accuracy
- Finally when the highest accuracy is achieved, we have our model ready

#### 4.3.2.3 Prediction on New Dataset

Now we have a list of test cases for which the values for factors are known. Based on the model designed above, we want the results whether to automate a particular test case or not. For this purpose, this new dataset is tested on the model designed above containing the factors which had lead to highest accuracy. Now, if the predicted value comes out to be say, more than 75%, it means this test case should be automated, else, it should not be automated.

## 4.4 Design of Framework

Every organization has its own way of doing things. Framework defines the standard of the organization, which is followed throughout the project.

### 4.4.1 Important Parameters for Developing Automation Framework

For designing a successful automation framework, there are some parameters to be kept in mind. These parameters are considered to design the framework to develop test scripts in OATS:

#### 4.4.1.1 Handle Data and Scripts Separately

The data input to the script i.e. test data should be kept separately and it should be inputted using external file. This is important because whenever any modifications in data will be there; there will be no need to make changes in the script. Just change to external data file will be sufficient.

#### 4.4.1.2 Function Libraries

All the reusable components and other functions like database connections, integration with other applications should be put in separate function libraries. It will reduce the redundancy and maintenance efforts in case of any changes required in the future.

#### 4.4.1.3 Maintenance and Extendibility

There are various changes in an application with every release. A good framework should support all the enhancements and it should allow modifying the existing features with minimal effort and maintenance cost. [25]

### 4.4.2 Types of Framework

An Automation Framework describes the set of guidelines which enforce some set of standards in order to make it easy to use for the users to work on it. There are various kind of automation frameworks, some are explained below:

**4.4.2.1 Keyword-Driven Framework:** It is a type of functional automation testing framework. It is also known as action word based testing or table-driven testing framework. In Keyword-driven framework, we use kind of table format, such as spreadsheet, in order to define the keywords or action words for each of the function that we want to execute.

**4.4.2.2 Data-Driven Framework:** This framework is based on creation of the test scripts, in which in place of using same hard-coded value, test data and/or input values are read from data files, each time the test runs. So, this way tester can test how various inputs are handled by the application effectively. It can be any of these data files- Data pools, Excel Files, ODBC Sources, ADO objects, CSV Files etc.

**4.4.2.3 Modular Framework:** The Modular framework is based on the concept of abstraction. It involves the creation of independent and individual scripts which represent the different modules of the AUT (application under test). These modules are used in a hierarchical manner to make large test cases. Hence, it builds an abstraction layer for the component to hide that particular component from rest of the application. So, if the changes are made to the some other part of the application, it does not affect that component. [26]

Out of use frameworks, Modular framework design is used for the research and implementation according to the requirements of the application.

### 4.4.3 Framework Design

Keeping in mind the parameters for an efficient framework, the framework is built using modular technology. The framework contains following assets:

- Databanks
- Object Repository
- Function Libraries

- **Databanks:** Databanks are used for parameterization purpose. Data in the form of .csv file or .txt file or database (SQL) is provided to the script. So, instead of using hard-coded values in the script, the values are fetched from an external file. So, if any change is required, there is no need to make changes in the script, just the changes to an external file can be made and the scripts will now fetch the modified values

- **Object Repository:** All the components on a webpage are recognized as the objects by the tool. The objects are identified by their object paths or X-paths. Object repository stores all the object paths on which the actions are to perform while playing back the scripts. The user-defined names can be given to all object paths for better understanding and these can be substituted in place of object paths in the script then.

  With release of new versions, sometimes the object paths of some objects are changed by the developers. If the object paths are in the common repository, then the changes are to be made at only one place instead of changing the all scripts. So, maintenance effort and cost is reduced this way

- **Function Library:** The functional libraries contain reusable functions, which can be used by different scripts. Scripts need to call these functions and pass the values as parameters. Use of functional libraries makes the framework modular. If the logic needs to changes in any function, the changes are required only at one place and now the scripts calling these functions will call the modified functions

## 4.5 Testing most Bug-Prone Areas of the Application

There are some areas in an application which are most bug-prone for e.g., transactions in accounting system. These areas must be tested more frequently than that of other non-bug prone areas to assure better testing of the system.

For this purpose, a small application is developed in Java which takes as input version, product and the area to show the bugs logged in the Bug DB corresponding to input values.

### 4.5.1 Design and Working

Bug DB is the central repository where the users log all the bugs found in the application corresponding to different version and products. The web application designed for testing most bug prone areas of application works as follows:

- It takes three inputs from user:
    - Release: The release no. of the product. For e.g.: 13.1, 14.0, 14.1.1
    - Product: The name of the product to be tested. For e.g. RWMS, RMS, POS, SIM etc.
    - Areas: An application or software always has different functional areas or keywords. It takes name of area to be tested for bugs. For e.g.: container, item, appointment etc.
- When user submits these three inputs, the application connects to the database of Bug DB and fetches the count of the bugs logged corresponding to entered release, product and area and shows it to user as "n Bugs Raised"
- Along with the count, a link is given to show the bugs logged. Clicking on this link shows all the bugs logged which are fetched from Bug DB as a list
- Now user wants to execute the test cases corresponding to all these bugs. For this purpose, a hash map is maintained which stores the 'area' as key and 'names of the test cases' as values.

- User is given a link to show the list of test cases corresponding to the area of the bugs. The names are fetched from the hash map and shown to the user as a list

- A button to execute these test cases is provided at the end to run the listed test cases as a batch. OpenScript tool takes the names of the test cases, treats them as aliases, execute these one by one and show the results to the user at the end

### 4.5.2 Benefits

- This application shows the count of the bugs corresponding to different areas of the application logged in Bug DB

- The areas having more count of the bugs can be segregated and tested more frequently

- The bugs can be identified and removed more fast and efficiently from the bug prone areas and hence giving more return on investment to the testing team

## 4.6 Results Reporting

The idea presented here is to generate and send the automated email with result attachments and send it to key stakeholders post the execution of batch finishes

### 4.6.1 Design

The mail script is devised to have 4 parameters array of Recepients, Subject String, MessageBody string and Attachment filepath.

The script would be coded using OpenScript and would utilize the capabilities of mail.java function library built on javaxmail.jar (available as a part of http://www.oracle.com/technetwork/java/javaee/downloads/java-ee-sdk-7-downloads-1956236.html.) [26]

### 4.6.2 Calling the Email Functionality from the Test Script

This is how the script will call the mailing script:

Library_name.function_name(list_of_receptients, subject_line, message_body, names_of_scripts);

For example,

Common_Lib.sendReport (mail_lists, subject, body, script_list);

- each sendReport will send one email based on the arguments.
- the "sendReport" is the name of the function implementing email functionality present in the Common_Lib
- mail_lists = a list of one or more email distribution lists. Each list consists of recipient's names/ids.
- subject = a text string containing the AUT and Functional Area tested, and a Pass or Fail status. An example is "WMS: Foundation Data Phase 1 – PASS".
- body = a plain text string.
- script_list = OpenScript Test script names whose results will be mailed out.

## Chapter Summary

The solutions to the problem discussed in previous chapter have been explained in this chapter. The application under test (AUT) on which all the testing is done is explained first. Then the comparative study is done between two testing tools HP QTP and OATS. Then the design of framework is presented for test automation to be implemented in OATS. An algorithm design is explained which will predict whether or not to automate the test case. Then the design of an application is described to test the most bug prone areas frequently

# Chapter 5

# Implementation and Results

## 5.1 Choice of Test Automation Tool

An average score is calculated for each factor discussed in the previous chapter to draw the graph showing comparison between both the tools

**(i) Utility**

Score for OpenScript: (1+4+4+2)/4 = 2.7

Score for QTP: (4+5+2+5)/4 = 4

**(ii) Ease of Learning**

Score for OpenScript: 3

Score for QTP: 4

**(iii) User – Friendliness**

Score for OpenScript: (4+5+4)/3 = 4.3

Score for QTP: (3+2+4)/3 = 3

**(iv) Scripting Features**

Score for OpenScript: (5+3)/2 = 4

Score for QTP: (1+5)/2 = 3

**(v) Recording Features**

Score for OpenScript: (3+4+4+3)/4 = 3.7

Score for QTP: (0+0+4+5)/4 = 2.2

**(vi) Ease of Debugging**

Score for OpenScript: (5+4+2)/3 = 3.7

Score for QTP: (5+4+4)/3 = 4.3

**(vii) Data Driven Testing**

Score for OpenScript: (5+5)/2 = 5

Score for QTP: (5+5)/2 = 5

**(viii) Test Results Reporting**

Score for OpenScript: (5+5+4)/3 = 4.7

Score for QTP: (5+5+4)/3 = 4.7

**(ix) Reusability & Maintenance**

Score for OpenScript: (5+4+5+4)/4 = 4.5

Score for QTP: (3+2+2+3)/4 = 2.5

**(x) Cost**

Score for OpenScript: 5

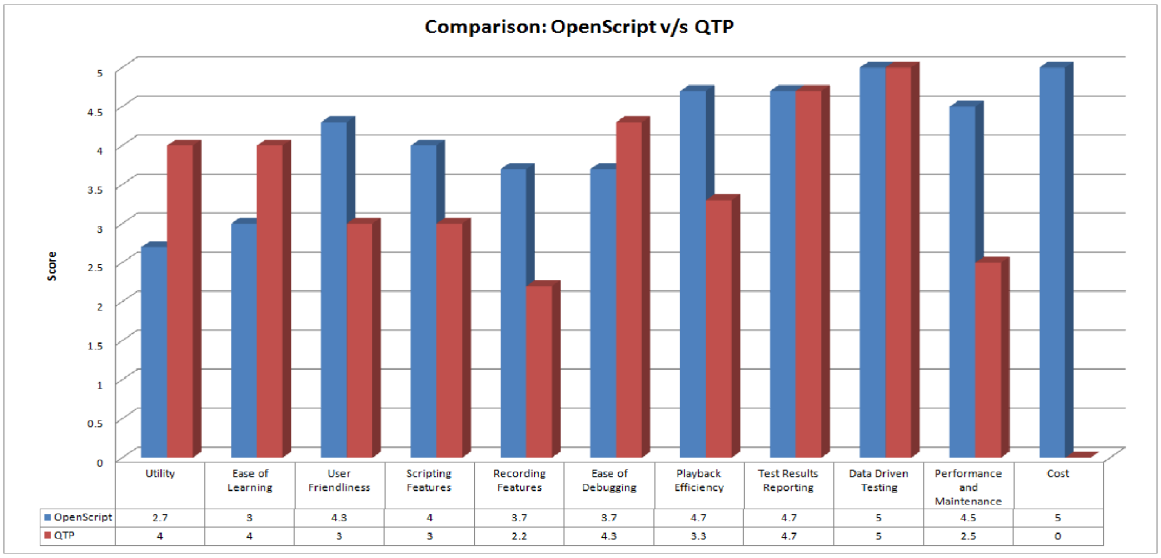
Score for QTP: 0

**(xi) Playback Efficiency**

Score for OpenScript: (5+4+4)/3 = 4.7

Score for QTP: (4+3+3)/3 = 3.3

### 5.1.1 Comparison Graph




| | Utility | Ease of Learning | User Friendliness | Scripting Features | Recording Features | Ease of Debugging | Playback Efficiency | Test Results Reporting | Data Driven Testing | Performance and Maintenance | Cost |
|---|---|---|---|---|---|---|---|---|---|---|---|
| OpenScript | 2.7 | 3 | 4.3 | 4 | 3.7 | 3.7 | 4.7 | 4.7 | 5 | 4.5 | 5 |
| QTP | 4 | 4 | 3 | 3 | 2.2 | 4.3 | 3.3 | 4.7 | 5 | 2.5 | 0 |

**Fig. 5.1: Comparison Graph between OpenScript and QTP**

## 5.2 Test Case Selection in Test Suite

### 5.2.1 Historical Data

We have a sample dataset containing data of 100 test cases. All the parameters are listed along with each test case. A value of 0 (low) or 1 (high) is assigned to each parameter. And finally we have the outcome that whether this test case was automated (1) or not automated (0) depending on these parameters. Below is the snapshot of sample dataset sheet.

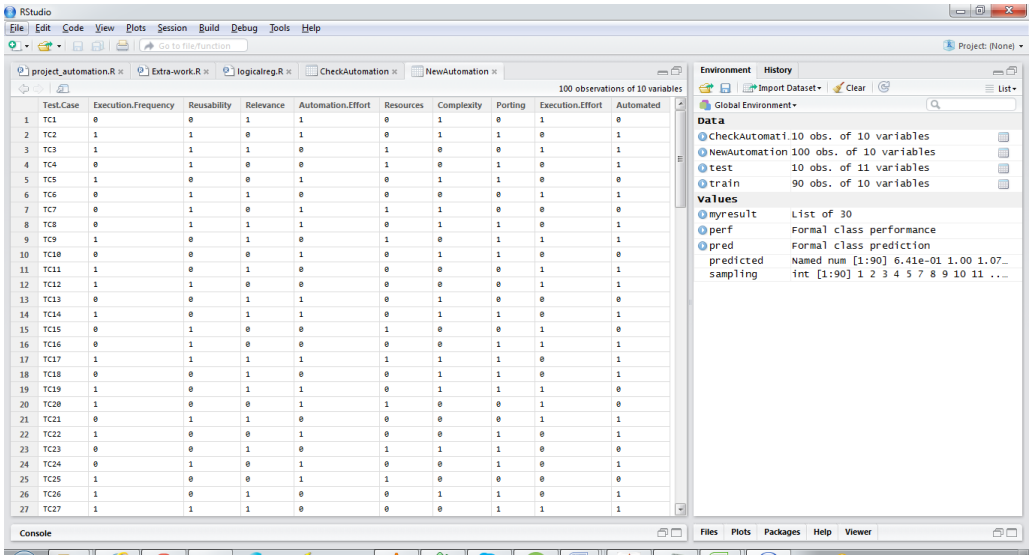



**Fig. 5.2 : Historical Data**

Here, the ideal factors that lead to the fact that test case should be automated are:

- Execution Frequency should be more (1)
- Reusability should be there (1)
- Relevance should be more (1)
- Automation Effort should be less (0)
- Resources needed should be less (0)
- Complexity should be less (0)
- Porting should be easy (1)
- Execution Effort should be less (0)

### 5.2.2 Implementation

The algorithm for predicting whether test case should be automated or not has been implemented using 'R', which is a language for statistical computing. This language is used for data mining and data analysis purposes

**Step 1:** Load the sample dataset of 100 test cases

*NewAutomation<-read.csv("F:/Oracle/M.TechResearch/Implementation/project/NewSampleData.csv",header=TRUE, sep=",")*

NewAutomation is the user defined variable in which sample dataset is loaded

**Step 2:** For visualization, dataset is plotted highlighting how many test cases are automated and how many are not

*plot(NewAutomation$Automated)*

We get the following plot as its result:

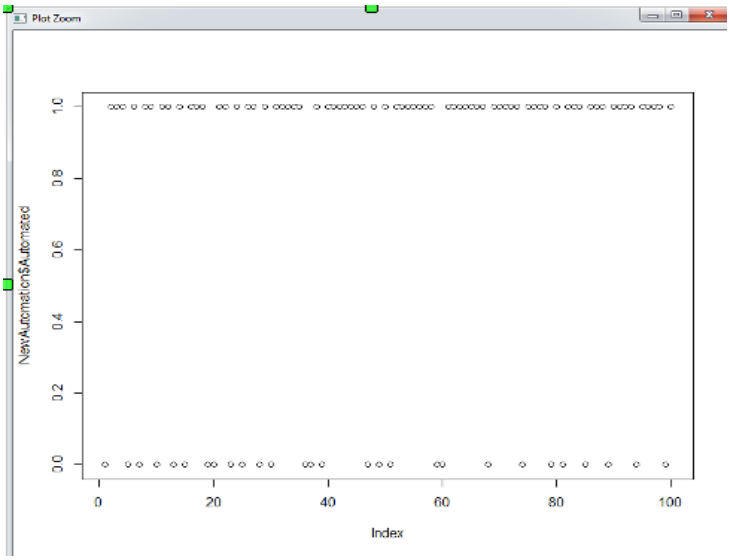



**Fig. 5.3 : Plot of Historical Data**

**Step 3:** Sampling of data

To check the accuracy of input data, sampling of data will be done. For this, out of 100 test cases data, 90 will be taken as sample and another 10 will be taken as the test data

*sampling<-sort(sample(nrow(NewAutomation), nrow(NewAutomation)*.9))*

*train<-NewAutomation[sampling,]*

*test<-NewAutomation[-sampling,]*

'train' variable points to training data and 'test' variable points to test data

*myresult<glm(data=train,Automated~Complexity+Reusability+Automation.Effort+Resources+Porting,family=binomial)*

**Step 4:** Now apply prediction on test dataset. If predicted value comes out to be more than 75% then the result is 1 else 0

*test$predValue<-ifelse(predict(myresult,newdata=test,type="response")>0.75,1,0)*

We get following predicted values as the result:

```
   Test.Case Execution.Frequency Reusability Relevance Automation.Effort Resources
19      TC19                   1           0         1                 1         0
34      TC34                   0           1         0                 1         0
42      TC42                   0           1         0                 0         0
45      TC45                   1           0         1                 0         1
59      TC59                   0           1         0                 1         0
60      TC60                   0           0         0                 0         1
64      TC64                   0           1         1                 1         0
82      TC82                   1           1         1                 0         0
92      TC92                   1           1         1                 0         1
98      TC98                   1           1         1                 0         1
   Complexity Porting Execution.Effort Automated predValue
19          1       1                1         0         0
34          0       1                0         1         1
42          0       0                0         1         1
45          0       1                0         1         1
59          1       0                1         0         0
60          1       0                1         0         0
64          0       0                0         1         1
82          0       1                1         1         1
92          0       1                0         1         1
98          0       1                1         1         1
>
```

**Fig. 5.4 : Output Predicted Values of Test Dataset**

**Step 5:** Check the accuracy whether the predicted data same as the actual output data which we have already by drawing the confusion matrix

*confusionMatrix(test$Automated,test$predValue)*

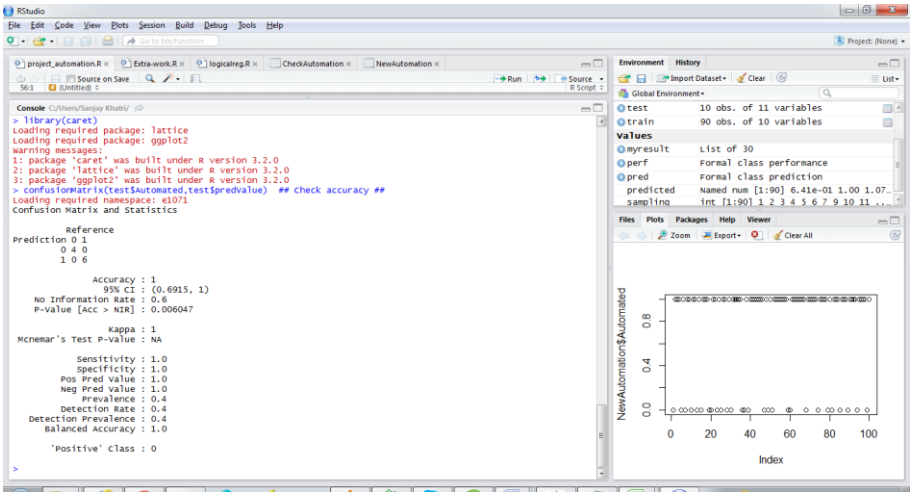



**Fig. 5.5: Confusion Matrix**

**Step 6:** Now for prediction for new dataset, load the new dataset

*CheckAutomation<read.csv("F:/Oracle/M.TechResearch/Implementation/project/CheckData.csv",header=TRUE, sep=",")*

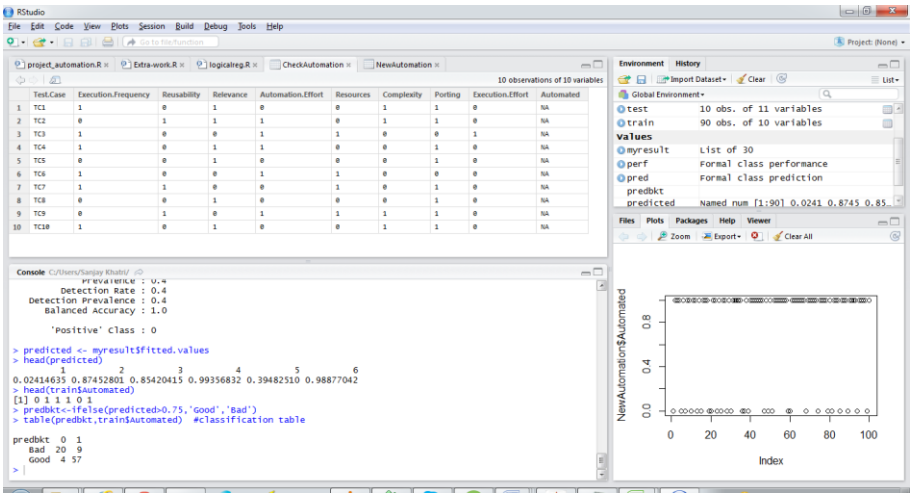


**Fig. 5.6: Test Data Set to be tested**

CheckAutomation is the variable in which new data set is loaded

**Step 7:** Now historical dataset is taken and model which we have made is applied to it

*myresult1<glm(data=NewAutomation,Automated~Complexity+Reusability+Automation.Effort+Resources+Porting,family=binomial)*

*CheckAutomation$Automated<ifelse(predict(myresult1,newdata=CheckAutomation,type="response")>0.75,1,0)*

Here we get the predicted values for new dataset that whether to automate the test case or not based on the historical data

project_automation.R × | NewAutomation × | CheckAutomation ×

Filter

| | Test.Case | Execution.Frequency | Reusability | Relevance | Automation.Effort | Resources | Complexity | Porting | Execution.Effort | Automated |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | TC1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 2 | TC2 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | TC3 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 4 | TC4 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 5 | TC5 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 6 | TC6 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 7 | TC7 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 8 | TC8 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 9 | TC9 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 |
| 10 | TC10 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |

**Fig. 5.7: Output Predicted Values**

## 5.3 Design of Framework

To overcome the drawbacks of record and playback testing, a framework is used. Following picture represents the folder structure followed for the framework implementation:

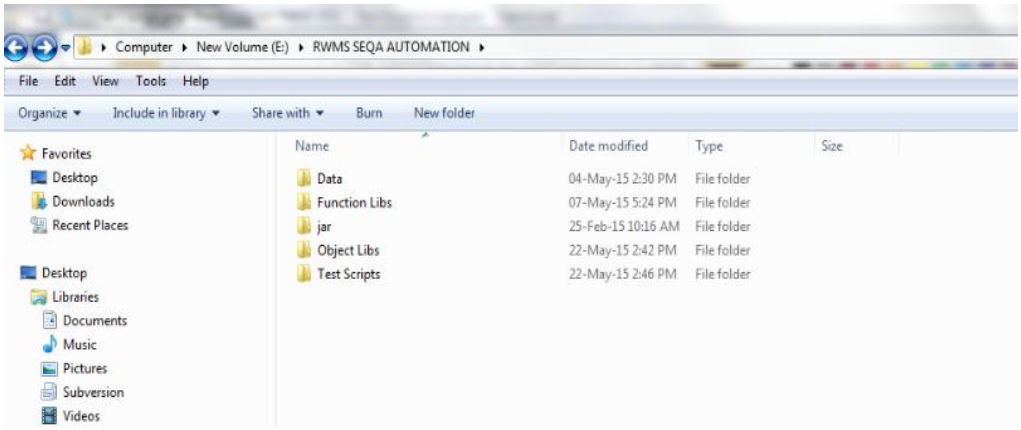



**Fig. 5.8 : Folder Hierarchy**

The implementation of framework covers following things:

### 5.3.1 Test Scripts

Test Scripts are the piece of code which contain the actual steps to be followed to execute a particular test case as shown in following figure:

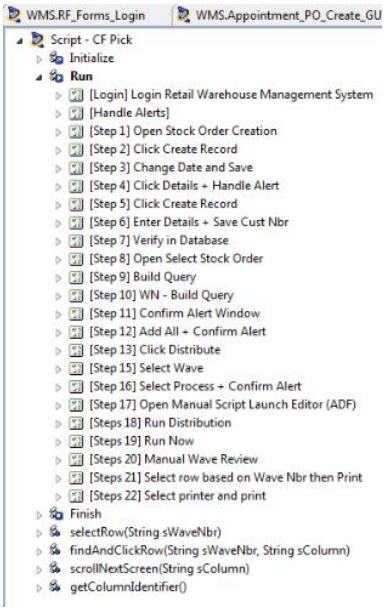
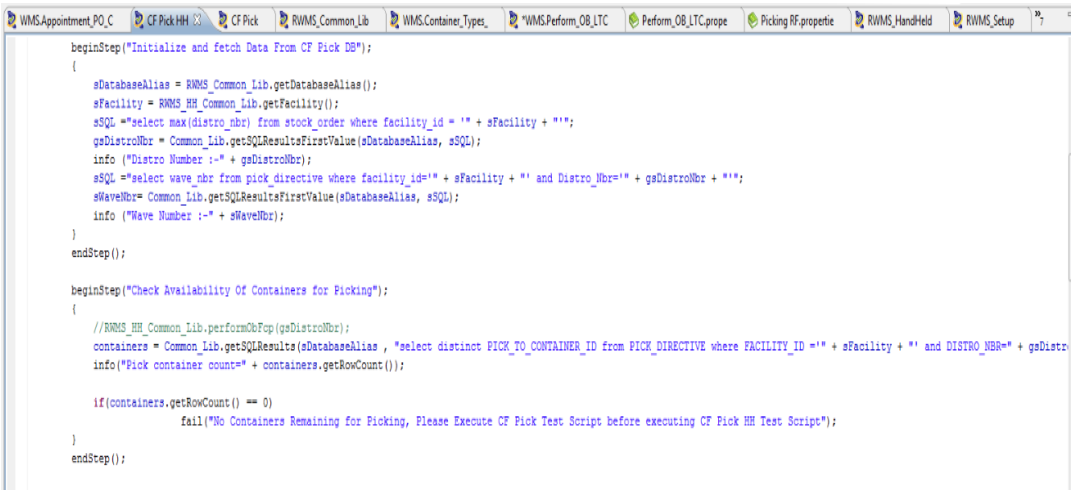



**Fig. 5.9: Test Case Steps**

To implement the framework, some files are attached with each script, these attached files are called 'Assets'. Assets can include databanks, jar files, object repositories, function libraries or sometimes other scripts (child scripts). A script view is shown below which shows the assets attached to a script:


**Fig. 5.10: Assets**

Inside the script, different steps are performed like fecthing values from data file in the variables, communicating with database, calling functions from function libs etc. Following figure gives the brief idea what all things are done in the test script:



```
beginStep("Initialize and fetch Data From CF Pick DB");
{
    sDatabaseAlias = RWMS_Common_Lib.getDatabaseAlias();
    sFacility = RWMS_HH_Common_Lib.getFacility();
    sSQL ="select max(distro_nbr) from stock_order where facility_id = '" + sFacility + "'";
    gsDistroNbr = Common_Lib.getSQLResultsFirstValue(sDatabaseAlias, sSQL);
    info ("Distro Number :-" + gsDistroNbr);
    sSQL ="select wave_nbr from pick_directive where facility_id='" + sFacility + "' and Distro_Nbr='" + gsDistroNbr + "'";
    sWaveNbr= Common_Lib.getSQLResultsFirstValue(sDatabaseAlias, sSQL);
    info ("Wave Number :-" + sWaveNbr);
}
endStep();

beginStep("Check Availability Of Containers for Picking");
{
    //RWMS_HH_Common_Lib.performObFop(gsDistroNbr);
    containers = Common_Lib.getSQLResults(sDatabaseAlias , "select distinct PICK_TO_CONTAINER_ID from PICK_DIRECTIVE where FACILITY_ID ='" + sFacility + "' and DISTRO_NBR=" + gsDistr
    info("Pick container count=" + containers.getRowCount());

    if(containers.getRowCount() == 0)
                fail("No Containers Remaining for Picking, Please Execute CF Pick Test Script before executing CF Pick HH Test Script");
}
endStep();
```

**Fig. 5.11 : Test Script Coding**

### 5.3.2 Data

To provide the parameterized data to the script, csv file is used, where first row shows the variable names and second row shows the variable values separated by comma.

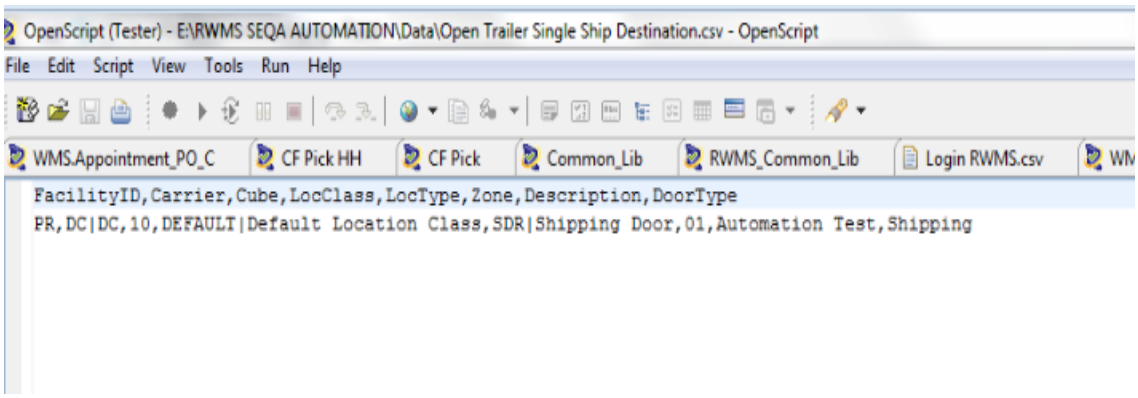



**Fig. 5.12: Test Data File**

Then to use this data in the script, this databank is loaded first with its alias name and then the value of the variable is fetched in some variable as shown below:

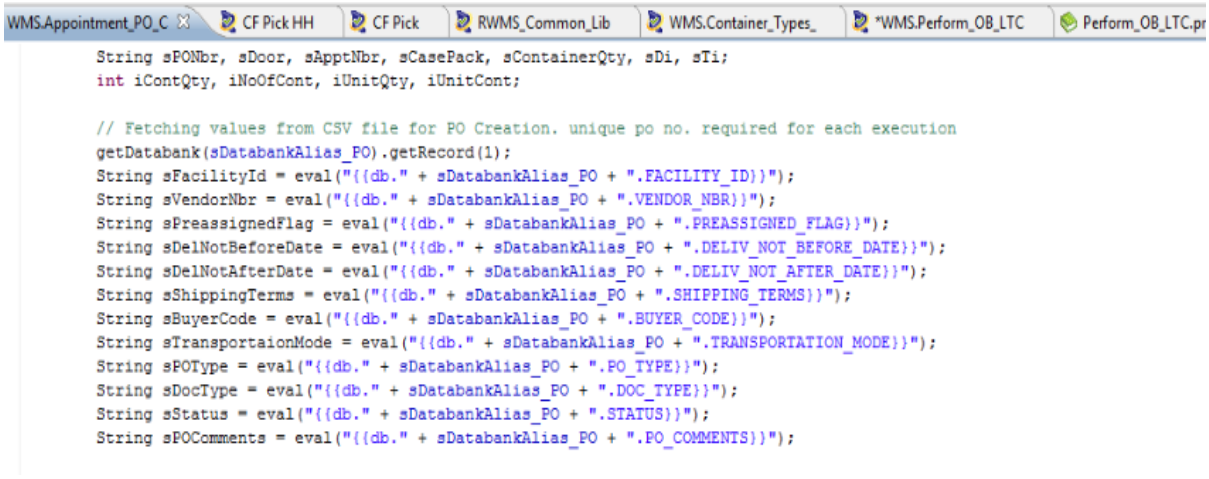



**Fig. 5.13 : Using Test data in script**

### 5.3.3 Function Library

This is the heart of the framework implementation. All the reusable code lies in the function libs. Use of function libs lead to reduced redundancy.

We have different libs according to the requirements. The generic functions which all the scripts will be using irrespective of the product; for eg; Execute Database Commands, Get SQL Results, Validate values etc. are placed in a Common Library as shown:


**Fig. 5.14: Functions in Function Library**

Then there are the libs which are product specific. So product specific functions for eg.: Login, Navigation, Click Link etc. are placed in this library as shown:

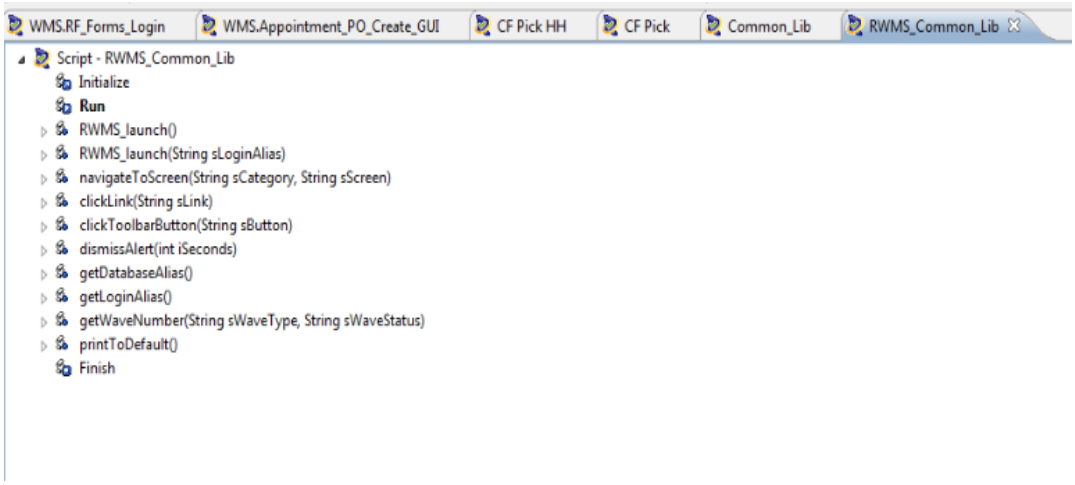



**Fig. 5.15 : Product specific function library**

Then further categorization of the libraries is done based on the application functional areas for eg: application area- Appointment will have different library and application area- Setup will have different as shown:

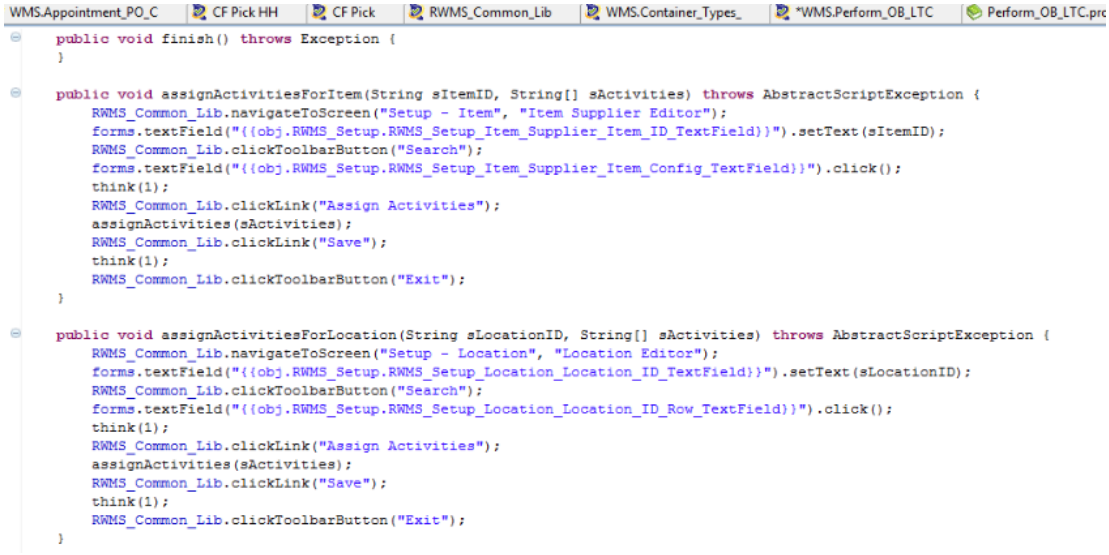



**Fig. 5.16 : Function Area specific function library**

### 5.3.4 Object Repository

Objects on UI are identified by their paths when playing back the script for testing. These paths are stored in the object repository as shown:

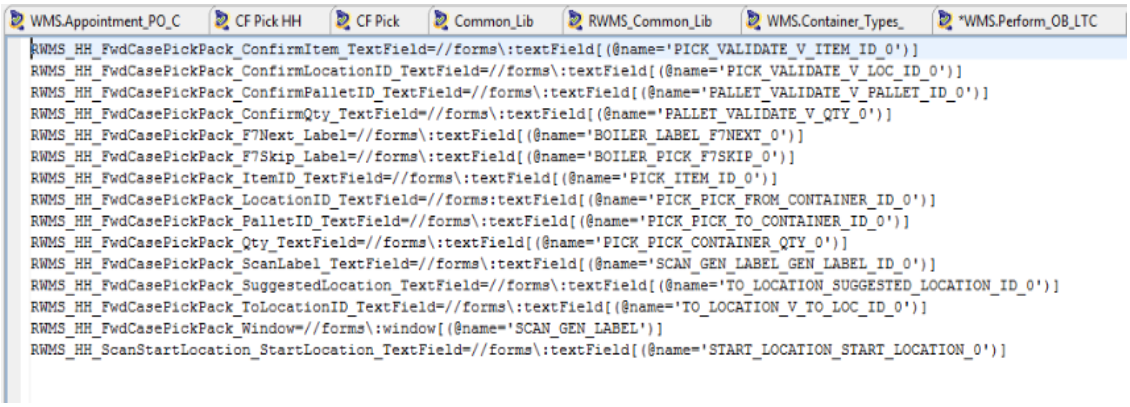



**Fig. 5.17 : Object Repository**

As shown, the object paths are stored in the user defined vaiables and now these variables are used in the script or functions for whatever action we want to perform on these objects

The figure below shows how actions are performed on these objects in the script:

```
public void assignActivitiesForItem(String sItemID, String[] sActivities) throws AbstractScriptException {
    RWMS_Common_Lib.navigateToScreen("Setup - Item", "Item Supplier Editor");
    forms.textField("{{obj.RWMS_Setup.RWMS_Setup_Item_Supplier_Item_ID_TextField}}").setText(sItemID);
    RWMS_Common_Lib.clickToolbarButton("Search");
    forms.textField("{{obj.RWMS_Setup.RWMS_Setup_Item_Supplier_Item_Config_TextField}}").click();
```

**Fig. 5.18: Performing Actions on objects**

## 5.4 Testing most Bug-Prone Areas of the Application

BugDB is the application, where customers or end users log the bugs encountered by them in the application. Following figure shows how BugDB UI looks:

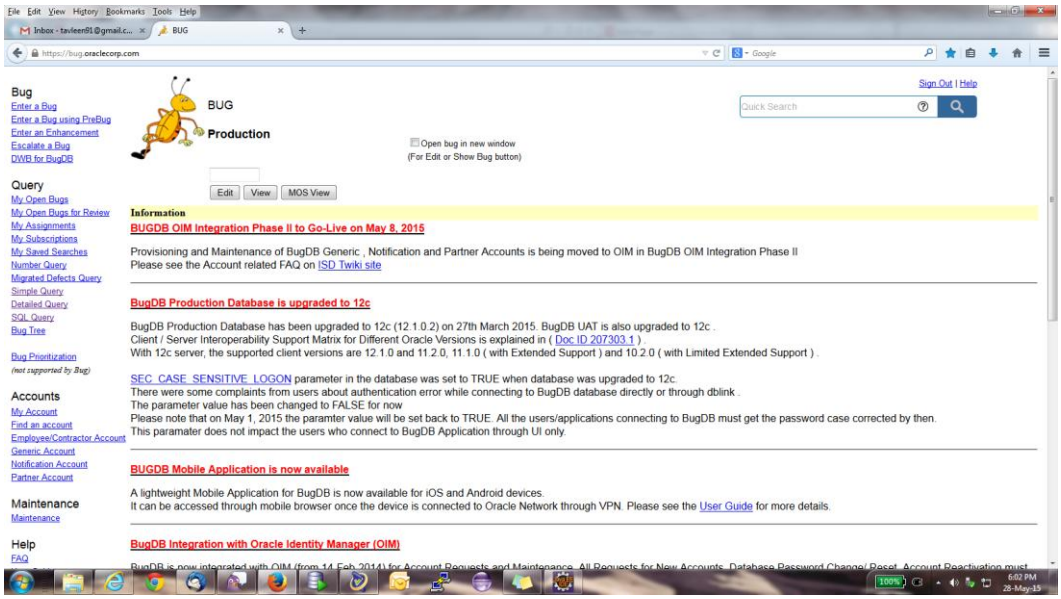



**Fig. 5.19: BugDB Home Page**

So, to test bug prone areas of the application most frequently, a java based utility is designed, which looks like following:

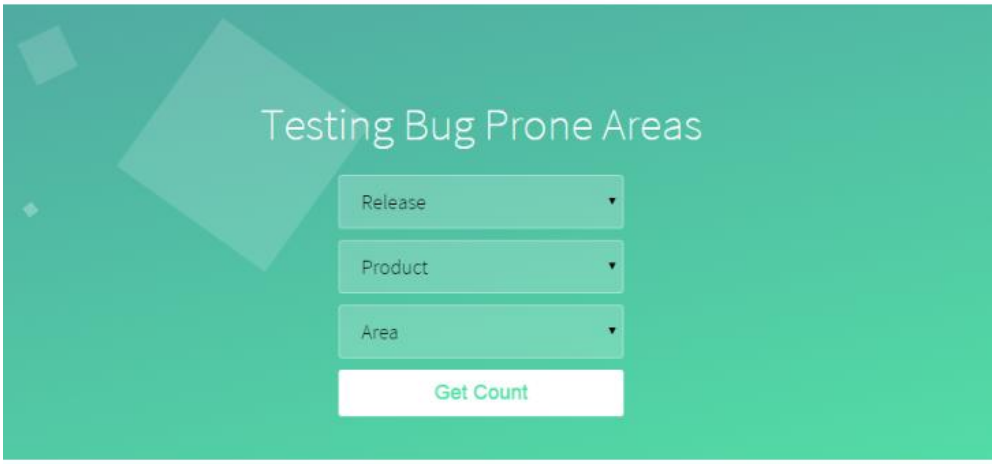



**Fig. 5.20: Application Home Page**

It has 3 dropdowns:

- **Release :** Every release of the product has some Release No., which is to be selected in this dropdown as shown:

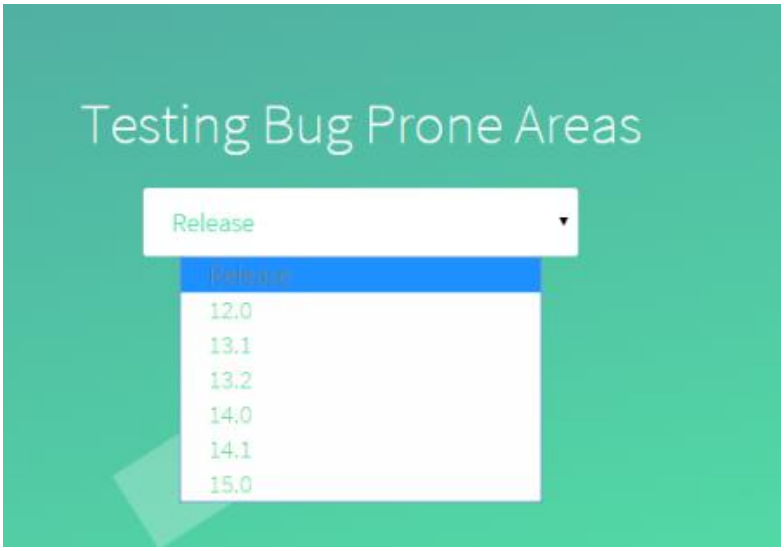



**Fig. 5.21: Selecting 'Release' Option**

- **Product :** An organization deals in many products. Which product needs to tested has to be chosen from this dropdown

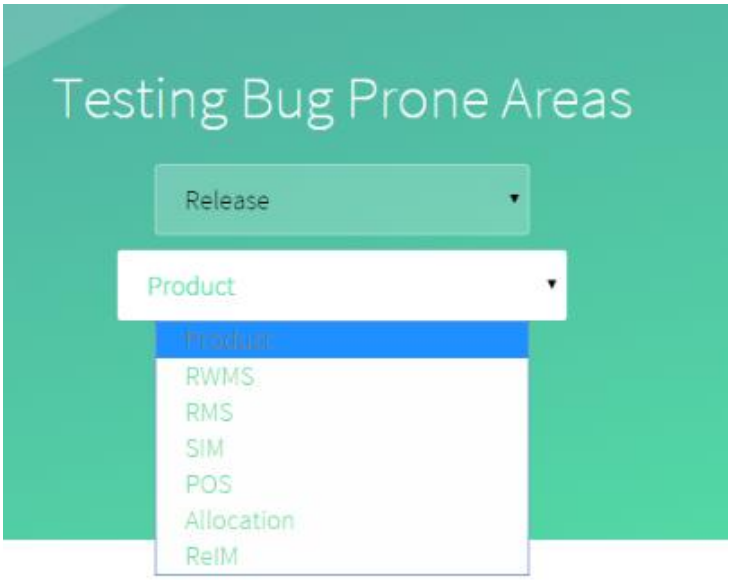



**Fig. 5.22: Selecting 'Product' Option**

- **Area :** An application or product further may have different functional areas according to its functionality. To find which functional area has how many bug counts, we will need to choose the area from this dropdown one by one as shown:

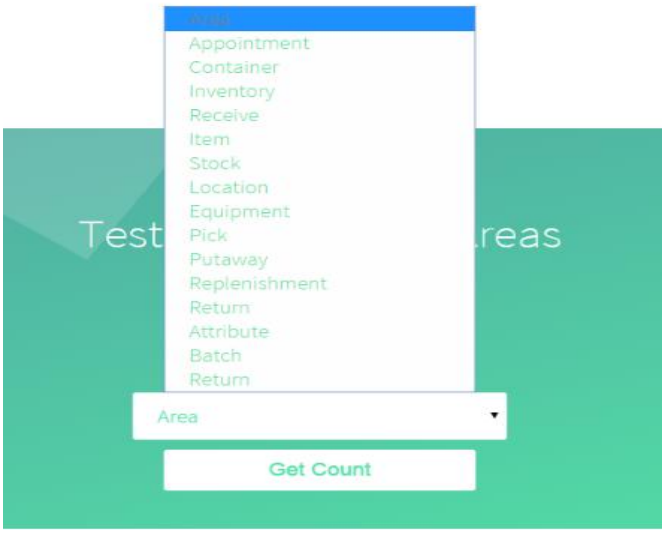



**Fig. 5.23: Selecting 'Area' Option**

Now once all the appropriate options are chosen, click on the 'Get Count'. Clicking on this button will fire a SQL query at the backend, which will get connected to the BugDB database and runs the query:

*"select count(*) from rpthead where utility_version='"+release+"' and product_name='"+product+"' and subject like upper('%"+area+"%')"*

Where,

'rpthead' is the table name in Bug DB database, which stores all the information regarding bugs.

'release' is the version number entered by user

'product' is the name of the product entered by user

'area' is the name of area in application entered by user

This query will give the count based on parameters chosen by the user and display it as shown:

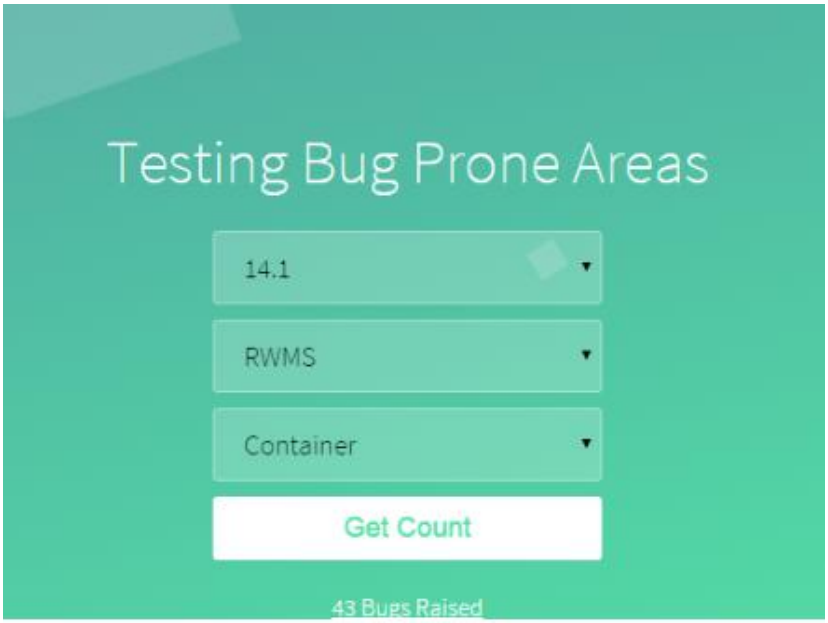



**Fig. 5.24: Bug Count Output**

Now, after getting the count, we want to see which bugs are raised. For this, click on this link of 'n Bugs Raised'. It will open the list of bugs on next screen as shown:

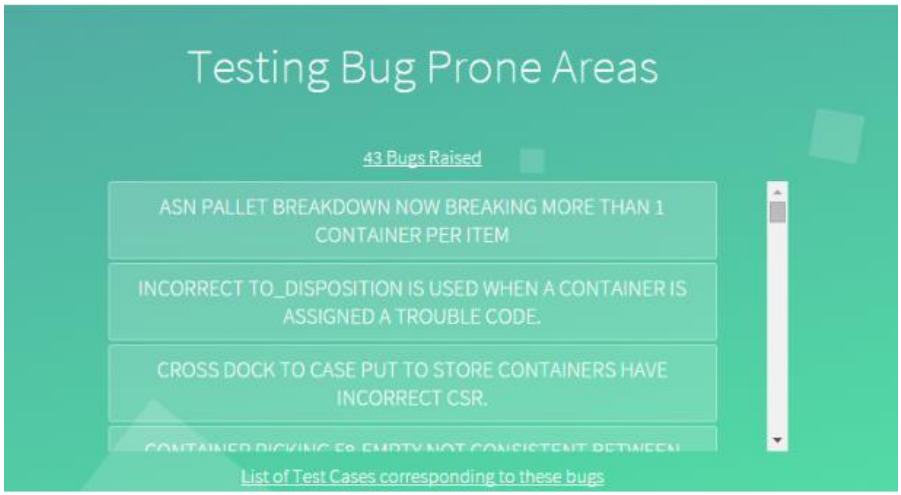



**Fig. 5.25: List of Bugs**

Now user wants to run the test cases to test this functional area. For this, a hashmap is maintained at the backend, which contains the 'Area' as the key and 'Test Cases' as the value corresponding to the areas. Clicking on the link 'List of Test Cases Corresponding to these Bugs' will show the user a list of test cases according to the area chosen by the user as shown on the next screen :


**Fig. 5.26: List of test cases corresponding to bugs**

Clicking on 'Execute' will trigger the batch which will run these list of test cases and show the results to the user at the end.

## 5.5 Results Reporting

The Email Test Script is called at the end of the automation test script sequence. This Email Test Script will send the result reports to the concerned group of people automatically.

This script implements the mail API by Oracle that is available openly at Oracle website: http://www.oracle.com/technetwork/java/javaee/downloads/java-ee-sdk-7-downloads-1956236.html. This mail API contains following functions:

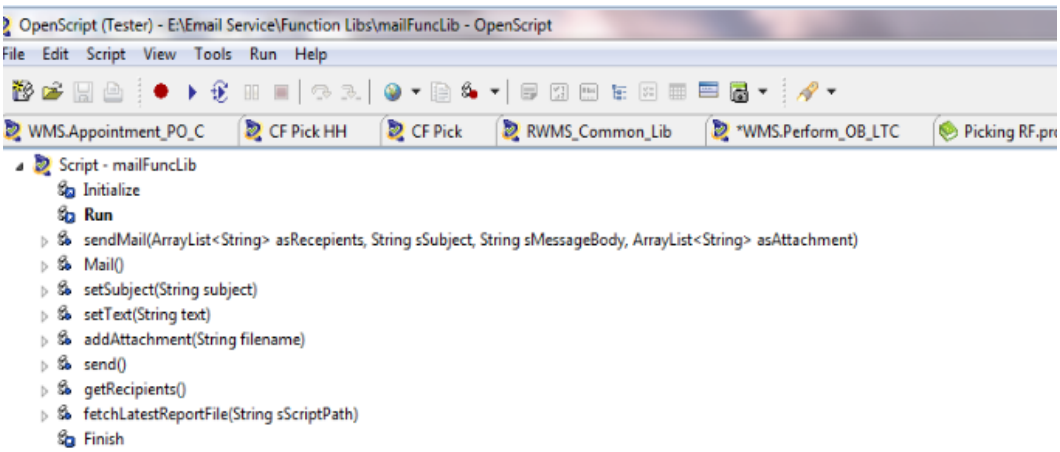



**Fig. 5.27: Email API**

And this is how the script looks, which uses this mail API:

```
public void run() throws Exception {


    String sScriptPath = "D:\\OpenScript Debug\\MailScriptDemo\\Test Scripts\\";

    String sSubject      = "Test #5 - change attachment name";
    String sMessageBody  = "Hi all,\r\n\r\nChange Attachment Name.  This is just a sample email from the Automation Email Service.\r\n" +
           "Please feel free to open the sample test result attachments in a browser.\r\n\r\n" +
           "Thanks";

    ArrayList <String> alRecipients = new ArrayList<String>();
    alRecipients.add("keith.ponnan@oracle.com");

    alRecipients.add("ben.faske@oracle.com");
    alRecipients.add("shiva.iyer@oracle.com");
    alRecipients.add("aashish.nadkar@oracle.com");
    alRecipients.add("ravikumar.nagaraja@oracle.com");
    alRecipients.add("cindy.metcalf@oracle.com");
    alRecipients.add("sushma.wadki@oracle.com");
    alRecipients.add("rgbu_automation_team_grp@oracle.com");
    alRecipients.add("nitin.ng.gupta@oracle.com");
    alRecipients.add("suresh.x.nataraj@oracle.com");
    alRecipients.add("keith.ponnan@oracle.com");
    alRecipients.add("almir.sehic@oracle.com");
    alRecipients.add("ben.faske@oracle.com");


    ArrayList <String> alScriptPath = new ArrayList<String>();
    alScriptPath.add(sScriptPath + "HelloWorld");
    alScriptPath.add(sScriptPath + "ParentScript");

    mailFuncLib.sendMail(alRecipients, sSubject, sMessageBody, alScriptPath);

}
```

**Fig. 5.28: Test Script calling Email functionality**

## Chapter Summary

This chapter presents all the results and snapshots, which are designed/implemented as the solution to the problem under research. The graph showing the comparison between HP QTP and OATS is presented. Then the implementation of framework in OATS is shown along with the snapshots. Then the output and results of the algorithm implemented in R is presented which predicts whether or not to automate the test case based on historical data. The snapshots of java based application implemented to test most bug prone areas of an application have been shown. Then at the last how email functionality is implemented to send automated mails is shown with the help of snapshots of the code.

# Chapter 6

# Conclusions and Future Scope

Software Testing is an important part of software development process. The aim of software testing is to find bugs and errors in the software. It also ensures that actual results are matching with the expected results and the requirements specified by the customer are being satisfied by the software developed. Software testing also tests the software for factors like efficiency, performance, response time, compatibility, security, reliability etc., and hence, ensures the quality of the software.

Various conclusions drawn out of this research work are discussed in this section.

## 6.1 Conclusions

This research work has been done to find out efficient methods and implement the framework, which result in improvement in performance, productivity and at the same time reducing the effort, resources and time taken by the Test Automation process.

(i) The first problem stands to the choice of correct test automation tool as discussed in the Section 4 problem statement chapter 3. This problem has been resolved by listing out the factors that should be considered when choosing the testing tool to test your product/application. As a POC, two available testing tools – HP QTP and OATS are compared based on these factors and a comparison graph is shown after the comparison

After conducting the comparison, there are several noteworthy conclusions:

- OATS is specifically for Web and apps developed using Oracle technologies. Hence, it cannot be used for every application in the market. But it is the best choice for testing Oracle technologies based applications. The other advantage is that it is Open Source; hence the cost is nil in terms of finance.

- QTP is more versatile tool since it supports many technologies, and can be used across many applications. Its efficiency is good, and it is user-friendly. The major drawback is the cost of the tool.

  So, one can conclude that both the tools have their own features. Hence, one should choose the right automation tool depending on the requirements of the application under test, expertise of tester in technology/language, budget etc. Selection of right tool in the early phase will help in reducing the testing cost, time and effort.

(ii) The next problem was to decide what all test cases should be included in the test suite according to whether they will be beneficial to be automated or not. This problem was solved by developing an algorithm in language 'R' which uses the concepts of Data Mining. We have a list of historical test cases, where based on certain factors a decision was taken manually that should these test cases be automated or not. The algorithm designed takes this historical data as input and design a model based on it and then new dataset which needs to be checked is applied on this model to predict whether these test cases will be beneficial to automate or not

(iii)The further problem discussed was one of the most important issue to be considered which directly affect the automation testing performance, resources utilization, maintenance etc. This is about "Framework". A well implemented framework can enhance the performance of automation testing to a great extent. So the solution for this problem is the implementation of modular framework in testing tool OATS. To implement the framework, function libraries are designed, where reusable code is placed, which all scripts will be using. For parameterization purposes, csv files are used, which will provide input test data to the script. To maintain all the object paths for a particular script in central repository, object libraries are used. This design of framework will help in maintaining modularity, hence reducing redundancy and maintaining efforts and thus, enhancing the performance of test automation process

(iv)The further problem discussed was to test most bug-prone areas of the application more frequently. As the solution to this problem, a java based utility is designed, which takes release, name of product and functional area of the application as input

parameters from user and get the bug counts by connecting to BugDB database. So, user can see the bug count of all the areas of the application and can decide which the most bug-prone area is and test that particular area. User can also see the bugs logged by user for the entered parameters and the list of test cases corresponding to this area. When user clicks on 'Execute', a batch containing these test cases in triggered which runs the test cases and show the results to the user at the end

(v)The further problem discussed was to communicate the results to the concerned group of people. To solve this problem, a script is generated, which uses mail API which is available freely to be used on Oracle's website. This script makes the process of sending results automated. When the sequence of scripts or a batch finishes the execution, it automatically attaches the results reports generated with the mail and send it to the concerned group of people

Therefore, all of these sub-solutions lead to the solution to main problem of enhancing performance and minimizing automation testing cost and time by implementing framework and using efficient methods.

## 6.2 Summary of Contributions

Following is the list of summary of contributions this research work makes:

- Parameters are decided based on which any testing tools can be compared for their performance and suitability for any application/product
- As an POC, two testing tools – HP QTP and OATS are compared
- A modular framework design for testing is proposed and same is implemented with testing tool OATS
- A java based utility which will test most bug prone areas of the application frequently is designed and tested
- A script is designed which uses Oracle Mail API to automate the mailing process which will attach result reports with the mail and send the mail to concerned group of people after execution gets completed

## 6.3 Future Research

There is always a scope of improvement in everything. So, a lot of research can be further done in the field of automation testing to improve its performance and reduce effort, resource utilization, cost and time. Some of the points on which work can be done are:

- Keyword based automation framework can be designed and implemented to further improve the performance of automation testing. In Keyword-driven framework, we use kind of table format, such as spreadsheet, in order to define the keywords or action words for each of the function that we want to execute. It basically gives more of tool independence and adds another layer of abstraction.

- One can go for Script less automation testing, which gives a lot of advantages over scripted automation testing. Script less automation testing is a new testing technique which makes the automation tool invisible. It acts as the interface between test automation tool and tester and allows performing automation using simple English. The underlying script less framework compiles in such a sense that automation tool understands this simple English. So, it gives user flexibility to change underlying automation tool whenever required without changing anything. So it gives the advantage of lowering down the maintenance efforts and cost.

## Chapter Summary

This chapter summarizes the conclusions drawn out of whole research work. It also lists out the summary of contributions done as the part of research work where the various algorithms and designs presented as the problem solution are listed out. Then it presents the future scope of the problem so that the researches picking up the same problem area can concentrate on these future scope points also

# References

[1] H. Tahbildar and B. Kalita, "Automated Software Test Data Generation: Direction of Research", International Journal Of Computer Science & Engineering Survey(IJCSES), Vol. 2, No. 1, Feb 2011, pp. 184-188.

[2] V. N. Maurya, R. Kumar, "Analytical Study on Manual vs. Automated Testing Using with Simplistic Cost Model", International Journal of Electronics and Electrical Engineering, Vol. 2, No. 1, Jan 2012, pp. 142-148.

[3] Vishawjyoti, Sachin Sharma, "Study and Analysis of Automation Testing Techniques", Journal of Global Research in Computer Science, Vol. 3, No. 12, Dec. 2012, pp. 36-43.

[4] A. Ieshin, M. Gerenko, V. Dmitriev, "Test Automation : Flexible Way", In: proc. of 5th IEEE Central and Eastern European Software Engineering Conference in Russia (CEE-SECR), Oct. 2009, pp. 249-252.

[5] M.M. Ali, T.K. Saha, "A proposed framework for full automation of software testing process", In: proc. of International Conference on Informatics, Electronics & Vision (ICIEV), May 2012, pp. 436-440.

[6] S Janardhano Rao, S. Ajay Kumar, R. Satya Prasad, K. Eswara Rao, "Quality Benefit Analysis of Software Automation Test", International Journal of Modern Engineering Research, Vol.2, Issue 5, Oct. 2012, pp. 3930-3933.

[7] John Kent, "Test Automation: From Record/Playback to Frameworks", In: Proc. of IEEE International Conference on Software Engineering, Vol. 2, July 2012, pp. 457-461.

[8] Jorge Oliveira, Cidinha Gouveia, "A Way of Improving Test Automation Cost-Effectiveness", In: Proc. Of IEEE 32nd Annual International Computer Software and Application Conference, Oct. 2008, pp.111-117.

[9] Vivek Motwani, "The When and How of Test Automation", In: Proc. of 3rd Annual International Software Testing Conference, May 2011, pp. 126-134.

[10] Test Automation Framework Development Challenges [online] Available: http://www.thoughtworks.com/insights/blog/guide-test-automation

[11] “Impact of Ineffective, Inefficient and Unsecure Test Data Data”, Compuware Corporation World Headquarters, Detroit, July 2013, [online] Available: http://www.compuware.com/assets/web-assets/pdf/MF_TestDataOptimization.pdf

[12] Gopinath Ganapathy, Deepa Vijay, “Towards Reduction of Cost of Software Quality”, Journal of Industrial and Intelligent Information, Vol. 2, No. 1, March 2014, pp. 63-66.

[13] Abha Jain, Manish Jain, Sunil Dhankar, “A Comparison of Automated testing Tools and their Impact on Software Testing”, International Journal of Engineering, Management and Sciences, Vol. 1, Issue 1, Jan 2014, pp. 8-12.

[14] Zeng Wandan, Jiang Ningkang, “Design and Implementation of a Web Application Automation Testing Framework”, In: proc. of IEEE 9th International Conference on Hybrid Intelligent Systems”, Vol. 2, Aug. 2009, pp. 316-318.

[15] John Kent, “Test Automation: From Record/Playback to Frameworks”, In: Proc. of IEEE International Conference on Software Engineering, Vol. 2, July 2012, pp. 457-461.

[16] M.J. Harrold, R. Gupta, M.L. Soffa, “A Methodology for Controlling the Size of a Test Suite”, In: Proc. of IEEE Conference on Software Maintenance, Vol. 3, Dec. 2009, pp. 302-310.

[17] M.M. Rosli, N.S. Mohammad, “The Design of a Software Fault Prone Application”, In: Proc. of IEEE Conference on Open Systems, Sept. 2011, pp. 338-343.

[18] Oracle, Retail Warehouse Management System (Dec. 2013) [online] Available: http://docs.oracle.com/cd/E12456_01/rwms/pdf/140/rwms-140-uiug.pdf

[19] Vienneau, R.L., “The Cost of Testing Software”, In: Proc. IEEE Conference on Reliability and Maintainability Symposium, 1991, Jan 1991, pp. 423-427.

[20] Richa Rattan, Shallu, “Performance Evaluation & Comparison of Software Testing Tool”, International Journal of Information and Computation Technology, Vol. 3, Issue 7, 2013, pp. 711-716.

[21] Xinbian Wang, Guangjun He, "The research of data-driven testing based on QTP", in Proc. of 9th International Conference on Computer Science & Education (ICCSE), Aug. 2014, pp. 1063 – 1066.

[22] Oracle Application Testing Suite [online] Available: http://www.oracle.com/technetwork/oem/app-quality-mgmt/s318966-hol-app-testing-suite-183607.pdf

[23] Oracle Application testing Suite for Oracle Applications [online] Available: http://www.oracle.com/technetwork/oem/app-test/ds-oft-acclerators-1626511.pdf

[24] A. Ilkhani, G. Abaee, "Extraction Test Cases using Data Mining; Reducing the Cost of Testing", In: proc. of 2010 IEEE International Conference on Computer Information Systems and Industrial Management Applications, Oct. 2010, pp. 620-625.

[25] Eun Ha Kim, Jong Chae Na, and Seok Moon Ryoo, "Implementing an Effective Test Automation Framework", 2009 33rd Annual IEEE International Computer Software and Applications Conference, Vol. 2, July 2009, pp. 534-538.

[26] Sidartha Gracias, Automation Test Frameworks [online] Available: http://www.cs.colorado.edu/~kena/classes/5828/s10/presentations/automation_test_frameworks.pdf

[27] Java Mail API [online] Available: http://www.oracle.com/technetwork/java/javaee/downloads/java-ee-sdk-7-downloads-1956236.html

# Research Publications

## Research Paper Accepted

- Tavleen Kaur, Shivani Goel, "Automated Testing Tools : QTP and OATS – An Execution Perspective", in 2015 IEEE International Symposium on Technology Management and Emerging Technologies (ISTMET 2015), Malaysia.

- Tavleen Kaur, Shivani Goel, "Automated Testing Tools : QTP and OATS – An Execution Perspective", in 7th International Conference on Cloud Computing Computer Science And Advances In Information Technology-ICCCCIT 2015

## Research Paper Communicated

- Tavleen Kaur, Shivani Goel, "Automated Testing Tools : QTP and OATS – An Execution Perspective", in 12th IEEE India International Conference (INDICON – 2015)

# You Tube Video Link

The URL of video uploaded on You Tube is:

http://youtu.be/rckzrYzN1lU