# Automation of Enterprise Product Integration Testing

A Thesis

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

**Master of Engineering**

in

**Computer Science and Engineering**

by

**Ashish**

(801632004)

Under the supervision of

**Mr. Shatrughan Modi**

Lecturer, CSED

THAPAR INSTITUTE

OF ENGINEERING & TECHNOLOGY

(Deemed to be University)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY

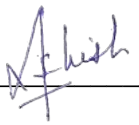
PATIALA - 147004

**July 2018**

# Certificate

I hereby certify that the work, which is being presented in the thesis, entitled **Automation of Enterprise Product Integration Testing**, in partial fulfillment of the requirements for the award of the degree of **Master Of Engineering** in Computer Science and Engineering submitted in Computer Science & Engineering Department of **Thapar Institute of Engineering and Technology**, Patiala is an authentic record of my own work carried out under the supervision of **Shatrughan Modi** and refers other researcher's work which are duly listed in the reference section.

The matter presented in this thesis has not been submitted elsewhere for the award of any other degree or diploma from any institution.

Signature: 

**Ashish**

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

Signature: 

**Shatrughan Modi**

Lecturer

CSED

# Acknowledgement

This research work cannot be marked as completed without acknowledging the ones who supported and guided me for the successful completion of this task.

It gives me enormous pleasure in showing courtesy and deep gratitude to **Mr. Puneet Saxena**, Senior Manager, Oracle RGBU, Minneapolis MN, US and **Mr. Advait Bhatt**, Quality Assurance Engineer, Oracle RGBU, Bangalore, India for their valuable guidance and mentorship that helped me to overcome every challenge I faced as I moved on in this work.

I wish to show my deep gratitude to **Shatrughan Modi**, Lecturer, Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala for his worthy guidance and continuous assistance throughout my work. He has always created a motivational and supported environment for me. His intelligence has always prompted me in doing more and more and this will surely give me advantage in future life.

I am grateful to **Dr. Maninder Singh**, Hon'ble Head of Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala for his kind support and providing basic access to technologies and the desired workspace.

I would also like to thank the institution, all faculty members of Computer Engineering Department, Thapar Institute of Engineering and Technology, Patiala for their special attention and suggestions towards this work.

**Ashish**

**801632004**

# Abstract

It is vital in any industry especially in IT to provide the right useful product to the desired customer with all the quality checks i.e. best quality one can ensure and that too at the right time. If we try to check the quality manually then a lot of time will be consumed as various input combinations has to be checked and it's also difficult to find all the defects because of human error. Moreover, it is costly, in term of resources, to manually analyze all results. These limitations were the main motivation behind this work so that we can overcome the limitations of manual testing on enterprise products using automation.

If we talk about manual testing, any change in the hardware configuration or a bit of change in source code leads to repetition of entire process. But in case of automation we have already pre-defined steps written through code that have to be followed and it is also very helpful in analyzing the results. Through automation we also gets a report for every run with all the details that can be used further for analysis purpose. One can also find the reason of failure of script through result that is generated through automation.

Enterprise Integration Testing (EIT) Tools and Utilities is what we came as a solution for automation. The major functionalities that comes under this are, EIT Tools, EIT Task Management, EIT Dashboard and EIT Reports with multiple utilities under each of them.

After using EIT Tools for regression testing, we came to know the most frequent reasons which were degrading the software quality and once these factors are highlighted then the preventive measures can be taken to improve the software quality.

# Table of Content

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 General Introduction to the Topic

It is vital in any industry especially in IT to provide the right useful product to the desired customer with all the quality checks i.e. best quality one can ensure and that too at the right time. If we try to check the quality manually then a lot of time will be consumed as various input combinations has to be checked and it's also difficult to find all the defects because of human error [1]. Moreover, it is costly, in term of resources, to manually analyze all results. These limitations were the main motivation behind this work so that we can overcome the limitations of manual testing on enterprise products using automation.

If we talk about manual testing, any change in the hardware configuration or a bit of change in source code leads to repetition of entire process. But in case of automation we have already pre-defined steps written through code that have to be followed and it is also very helpful in analyzing the results. Through automation we also gets a report for every run with all the details that can be used further for analysis purpose. One can also find the reason of failure of script through result that is generated through automation.

## 1.2 Need of Testing

The best quality of a product can be ensured through testing. Testing is the most crucial phase of development cycle. It helps in finding out more relevant information related to the quality of product and how it can be improved. It also checks for the flow with which the product should work. In all, testing can be used for giving a better vision to products.

Bugs finding, error findings and defect findings are the main objectives of testing which are not generally taken care in the development cycle. At development time, these are not desirable [19]. Testing techniques available for testing depends on the nature of testing like Integration testing, Quality Acceptance testing and Unit testing are some of them.

## 1.3 Different Types of Testing

Different types in which Software Testing can be classified are as follows [2],

- Security Testing
- Functional Testing
- Performance Testing
- Regression Testing
- User Acceptance Testing

Among the above mentioned testing methods, Regression Testing is most laborious, time consuming and error prone testing type as in it, whenever the system undergoes any kind of change then complete execution of all set of test cases are done. Hence, a term is introduced called as Test Automation which eases the task of Regression Testing.

### 1.3.1 Functional Testing

Functional testing is a type of black-box testing and a quality assurance (QA) process [3]. All the test cases are based on the specifications of the software component that is under test. Functions are generally tested by feeding them with input and then examining the output. Internal program structure is rarely considered in it. What the system does is checked in functional testing.

### 1.3.2 Regression Testing

Regression testing is major part of the integration testing. Whenever there is a change in some part of the system, then entire system undergoes testing to find any chances of failure. Regression testing comes out to be costly in terms of money and time if, complex application are considered.

The main goal of regression testing [4] is to make sure that with introduction of new changes the previous functionalities doesn't stop working in appropriate manner. It checks if the software quality is compromised with new changes or not. Test cases written for previous version of software can be for new version also. We must keep in mind that number of test cases should never grow with software.

Test suite selection techniques try to reduce the cost of testing by running a subset of the tests, such as those that execute the modified source code, in order to ensure that the updated program still operates correctly.

### 1.3.3 Performance Testing

Performance testing in software engineering is in general, a testing practice that is used to determine how system performs in terms of stability and responsiveness under a particular condition. It is also used to measure, verify, validate or investigate other quality attributes of the system, such as reliability, resource usage and scalability.

### 1.3.4 User Acceptance Testing

User acceptance testing is a test conducted to determine if the specification or contract are met with the told specifications. For acceptance testing we can perform performance test, chemical tests or physical tests based on the requirement. Black-box testing performed on a system prior to its delivery in systems engineering (for example: lots of manufactured mechanical parts, batches of chemical products, or a piece of software) [5].

### 1.3.5 Security Testing

Security testing is a way to identify flaws in the security mechanisms of the information system that is used to protect data and maintain functionality. As there are logical limitations in security testing thus passing security testing does not indicate that there are no flaws in the system but adequately satisfies the security requirements.

## 1.4 Motivation

Arrangement of tools, ideas and assumptions to provide some core functionalities was the main goal of test automation. Error recovering, monitoring, logging, functioning and reporting are the functionalities that it will have. There are different approaches to automation that use different methodology [6]. Automation mainly assign same job to computers and thus reduces the job of test engineers. If regression testing is to be done, then a person (manually) takes more time to execute test cases as he has to validate all the

attributes manually which takes a lot of time. The time taken in validations can be reduced if it's done through automation and more test cases can be executed on a particular day as described in Figure 1.1.

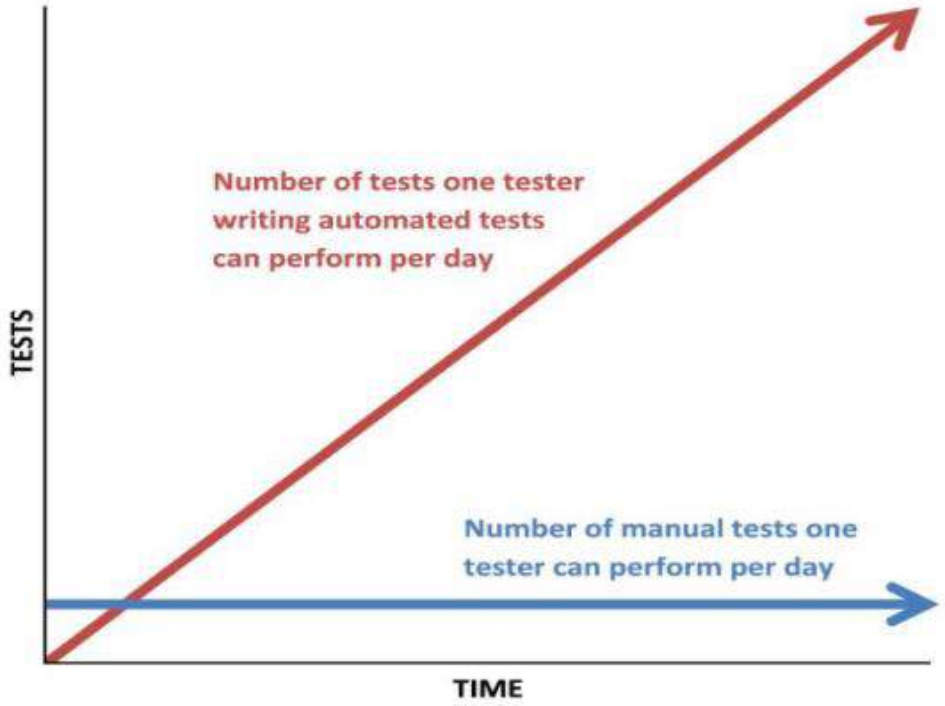



Figure 1.1 Comparison between Manual and Automation Testing

Database validations, load performance and compliance of requirements are what, that are included in automating manual testing process. New capabilities and features are also added with the growing demand of support in testing. Regression testing is preferable to find bug in the code and reduce risk in fastest way with less effort.

## 1.5 Problem Definition

Design, Development and Implementation of Automation Tool and Utilities which are used by Enterprise Integration Testing team for reducing manual efforts. As a total of 230+ scenarios are there that needed be tested regressively thus time plays a crucial role here. Manual Testing was not giving the desired result thus there was a need for introduction of Automated Testing that can regressively test 230+ scenarios efficiently.

## 1.6 Objectives

The main goal of this project is to automate functionalities which are regularly used in test execution in order to,

- Increasing productivity of manual testing process.
- Reduce cycle time of testing by automating functionalities and reducing efforts.
- Developing an easy to understand interface to manage the process.
- Improving consistency and quality in execution of test.
- Re-configuring the portal as per the availability of test environments, products and databases.

## 1.7 Thesis Organization

- **Chapter 1:** This chapter introduces basics of both manual and automated testing. It also introduces the basic methodology about how one can test the quality of software. It also describes different types of testing tools available for automation testing.
- **Chapter 2:** Describes the fundamentals required for this thesis. It describes the Significance of Validation Phase in the Integration Testing. It also describes the review of the existing Retail Applications so that a better Integration Testing can be achieved.
- **Chapter 3:** This chapter defines the Integration problem and the way by we are dealing it in the thesis. It also describes the main workflow.
- **Chapter 4:** This chapter explains the different tools and utilities that we have used for a better understanding of results and to achieve a better regression testing.
- **Chapter 5:** In this chapter the results of thesis are discussed and also the main reasons behind the not so good quality of applications are listed.
- **Chapter 6:** This chapter states the conclusion and the scope for future work.

# Chapter 2

# Literature Review

## 2.1 Automated Testing

Automation testing tools are used for automated testing. The main task of automation testing tool is to test the intended condition and reduce human efforts. It is much faster than manual testing and also more reliable as each test case is performed with precision. Software quality and reliability are increased with automation testing [7]. In automation testing we can write sophisticated test cases to detect hidden defects which is not possible in manual testing.

### 2.1.1 Testing Life Cycle

Testing life cycle may vary from organization to organization but, there is a typical cycle for testing [8]. The below cycle is used in organizations which uses the Waterfall development model. The same practices are also found in other development models, but might not be as clear or explicit.

- **Requirements analysis:** In the requirements phase itself, the testing of the software should began. In the design phase, two things can be determined namely, what aspects of a design are testable and with what parameters those tests work.
- **Test planning:** Test strategy, test plan, testbed creation. A plan is needed as number of activities will be carried out during testing.
- **Test development:** Test scenarios, test procedures, test scripts, test datasets, test cases to use in testing software.
- **Test execution:** Proper planning and test documentation is done and then testers execute the software and if there is any error they report to development team.
- **Test reporting:** Testers makes metrics and also the final reports after the testing is completed on their test effort and also ensure if the software is ready to be released or not.

- **Test result analysis:** Test result analysis or defect analysis, is done by the development team usually along with the client, so that we can ensure what defects needs to be fixed, assigned, deferred (left for dealing in future) or rejected (software is fit to use).
- **Defect Retesting:** Once defects are fixed by development team, the software is again tested for the proper functioning.
- **Regression testing:** It is common to have a small test program built of a subset of tests, for each integration of new, modified, or fixed software, in order to ensure that the latest delivery has not ruined anything and that the software product as a whole is still working correctly.
- **Test Closure:** Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons learned, results, logs, documents related to the project are archived and used as a reference for future projects.

### 2.1.2 Different Types of Automated Testing

- **Automated Web Testing:** Software testing that uses automation for focusing on web applications is called automated web testing. Number of issues can be addressed before the system is revealed to the public by complete testing of a web-based system before going live [21]. Issues can be operating systems, devices, and its ability to adapt to the multitude of desktops, its accessibility to handicapped users and fully able users, the basic functionality of the site and the security of the web application.
- **Automated GUI Testing:** Graphical user interface testing is the way of testing a product's graphical user interface to ensure it meets its specifications. Variety of test cases are used in this.
- **Automated Unit Testing:** Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors [9]. Depending on the organization's expectations for software development, unit testing might include static code analysis, data-flow

analysis, metrics analysis, peer code reviews, code coverage analysis and other software testing practices.

- **Automated Integration Testing:** Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system [10].

There are a number of open source testing tools [20, 23] available in the software market. Although the core functions of these tools are similar (depending on the automation case), they differ in functionality, features, usability [11]. The parameters that one looks for comparison of tools are,

- Recording Efficiency
- Capability of generation of scripts
- Data driven testing
- Test result reports
- Reusability
- Execution speed
- Playback of the scripts
- Easy to Learn
- Cost

## 2.2 Introduction to EIT

Enterprise Integration Tools and Utilities are used by EIT members to ease their integration work. The major functionalities that comes under this are, EIT Tools, EIT Task Management, EIT Dashboard and EIT Reports with multiple utilities under each of them. Graphical representation of analysis is shown in EIT Dashboard. Reporting tools for messages covers, product traceability matrix and foundation matrix are available under EIT Reports. Functions like Configuration Management, Data Verification and Transaction Validation lies under EIT Tools section.

## 2.3 Enterprise Products

If we don't know what is really happening behind the scene then it's totally of no use of automating the things as we don't know what should happen after what. So understanding the characteristics of automation as well as retail cycle together is very important. This will help in better analysis of result.

A brief overview of retail applications, test scenarios, framework and technologies are necessary for a good quality of automation. Automated test cases should be reusable and maintainable. Some of the retail applications that were used in this project are as follows,

- **RMS –** RMS stands for Retail Merchandising System [12]. It acts as the central repository for all the other applications in retail world i.e. if any kind of update is made in any application then it's obvious that change will be reflected in RMS. It act as an admin application that have access to functionality of all other retail applications available. It also makes sure that data is integrated by using the Retail Integration Bus. It provides a smooth environment so that different applications can communicate and helps in taking critical decisions. Oracle have dedicated applications for different retail tasks like Sales, Analysis, Warehousing and many more. RMS acts as a heart of all the above mentioned products.
- **SIM –** SIM stands for Store Inventory Management [13]. This application is basically used for keeping a track of stock of good in different stores. The stock changes after every task is performed like, Ordering, Returning and Transfers. The stock for the good changes accordingly and SIM keeps a track for that. In short it tracks shipping and receiving of goods.
- **WMS –** WMS stands for Warehouse Management System [14]. All the operations related to warehousing are taken care by this application. Some of the operations that are done in WMS are picking, shipping, receiving and block storage of goods. Every location (store) has a unique warehouse for it that stores the goods that cannot be kept in stores itself.
- **ReIM –** ReIM stands for Retail Invoice Matching [15]. Each Order and RTV (Return to Vendor) scenarios have invoice which consists for cost and quantity. The main task of ReIM is to verify that cost and quantity from the invoice. If we

find any mismatching in the cost and the quantity then it's sent for review and both cost and quantity are matched.

- **RPM –** RPM stands for Retail Price Management [16]. This application is used for setting the price for a particular item. It is also used to declare a sale or extra discount on the item. Promotions and Clearance are created on item by zone i.e. sale is declared in a particular area. The sale can be amount off or percent off.
- **Allocation –** Allocation [17] can be used for distributing items from different types of sources to a single location. It's a kind of transfer which includes shipping from multiple stores and receiving at single store. It can be initiated by transfer, warehouse, PO etc. Scheduled Allocations, What if Allocation and Standard Allocation are some kind of allocations.

## 2.4 RIB

A require a medium through which different retail applications can communicate and share data with each other. This feature is facilitated by Retail Integration Bus (RIB) [18]. RIB is there for all products like RMS, SIM, WMS etc. as each of them have to share data according to the scenario as in Figure 2.1. There are two adapters in this, namely, Publishers and Subscribers. Publishers are used for publishing the data onto the RIBs which will then be accessible to the application for which it was published. For publishing the data onto RIBs the Publish adapter must be up and working. Subscriber is used for subscribing i.e. taking the data from the RIBs. If one application has to take some data from the RIBs then its Subscriber adapter must be up and working.

The functionalities that are provided by RIBs include support for fast delivery of retail volumes, ability to fix and retry message, real time messaging, guaranteed sequential delivery within a message family regardless of errors, guaranteed once only delivery and multi-threading. Due to all of these an infrastructure that is highly available and highly reliable is created.

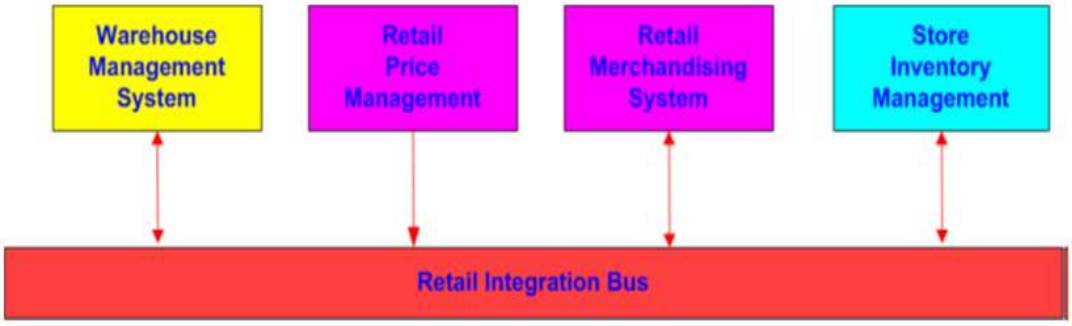



Figure 2.1 Retail Integration Bus

## 2.5 Foundation Data

Foundation Data is the pre required data that is needed before performing any scenario. This data can include information about Items, Warehouses, Suppliers and Stores. We need this data because for performing any scenario we must have some Stores, Warehouses, Suppliers and Items in hand or else we won't be able to perform any scenario. This data also include the information about the stocks at different places as well as different types of Stores and Items.

There are many changes as per the requirement in applications. So applications must be deployed regularly. Database and basic transactions should be tested after each releases. QA environments provides details about applications, integrations and databases.

| Supplier | Warehouse | Store | Item |
|---|---|---|---|
| • Supplier can be manufacturer of item<br>• Located in various countries | • Retailers like Big Bazar have their warehouse to keep stock of items in bulk<br>• Items are transferred to various stores | • Located in countries<br>• Various types of items | • Various kind of items are stored in stores<br>• Items are identified uniquely in stores |

Figure 2.2 Foundation Data

# Chapter 3

# Architecture and Workflow

## 3.1 Workflow to Achieve Objectives

The overall flow of the process start from executing test cases to different applications involved and the results can be seen from Figure 3.1.

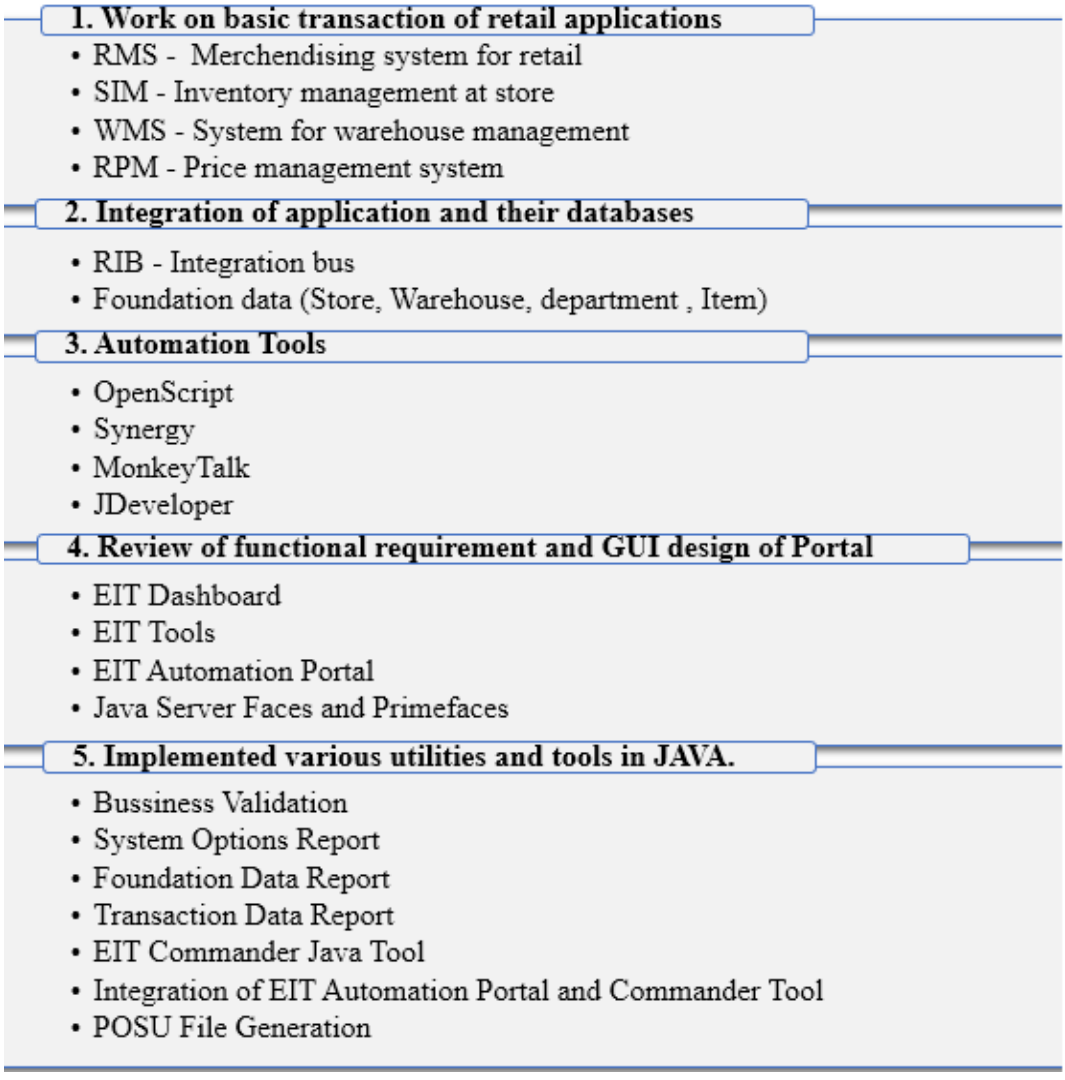



Figure 3.1 Overall Workflow

At very initial stage, a particular test case is executed manually for a better understanding of the scenario. Then while automating the case, the integration services (RIBs) are taken

into consideration. Sometimes due to one scenario the RIBs may get down thus proper precautions are taken. The tools that were used for automating the scripts depend on the nature of the scenario i.e. the different applications used in. For example, A Warehouse to Warehouse Transfer only need the RIBs (both Subscriber and Publisher) for both Warehouse and RMS to be up and working. Subscriber usually subscribes the data information from the RIB whereas Publishers publishes the data information into the RIB. In this case only OpenScript is used as Warehouse application is Form based application. If SIM is in the picture, suppose a Store to Store Transfer then one have to use OpenScript as well as MonkeyTalk and JDeveloper for automating this scenarios as SIM application is built in Mobile Application Framework. After the script is ran completely one can check the results from the EIT Dashboard which shows the exact reason for the script failure. After the cause of failure of script is known, the same is checked with Dev team and required steps are taken so that the software quality can be improved. The utilities that are created are used for the better and easy flow of the test execution. These utilities gives us the pre required data that is needed for the execution of the scenario.

## 3.2 Automation Scenario

OpenScript was used for the ADF application and forms, Synergy is used for java applications and MonkeyTalk for Mobile Application Framework applications. After each stage of the scenarios, database validations and product UI validations are performed to check the transaction is performed successfully or not.

Tool used for the automation of specific product:
RMS, WMS, Allocation, RPM – **OpenScript**
SIM – **Synergy**
SIM-MAF – **MonkeyTalk and JDeveloper**
For this thesis the following scenarios [12] has been executed,

- Purchase Order (PO)
- Direct Store Delivery (DSD)
- X-Order
- Item Request

- Transfer
- X-Transfer
- RUA
- Replenishment
- Return to Vendor (RTV)
- Commerce Anywhere (CA)
- Cross Channel (XC)
- RCA
- Mass Return Transfer (MRT)

Some of the scenarios are described below.

### 3.2.1 Purchase Order (PO)

PO is a scenario using which the order for the item by particular warehouse will be placed and inventory will be provided to that warehouse by the supplier. In this scenario, PO will be created in RMS and then in WMS appointment will be created with type PO and appointment will be received. There are different types based on the appointment received after edit, appointment deleted, PO cancelled. Products included in this scenario: RMS, SIM and WMS.

### 3.2.2 Direct Store Delivery (DSD)

DSD is a kind of purchase order in which warehouse doesn't come into picture. Once the order is created in RMS, the same is received directly in the destination store i.e. the quantity is shipped directly from supplier to the store.

### 3.2.3 Item Request

In Item Request scenario, item request is created in SIM which will basically trigger a PO for the item. So after creating item Request, the PO which is generated for that item should be received in SIM and validate in RMS and SIM UI and DB. There are different scenarios

of Item Request based on the item is having source as warehouse or supplier, Item Request with UOM case or unit. Products included in this scenario: RMS and SIM.

### 3.2.4 Transfer

In transfer scenario, the item will be transferred from one location to the other so quantity of item will be decreased at one location and will be increased at other. There are total four types of transfers:

- **Store to Store** - The transfer can be RMS initiated or SIM initiated that is it can be created in RMS or SIM. Source and Destination of transfer will be store. Transfer will be shipped from source store and it will be received in destination store in SIM. Products included in this scenario: RMS and SIM.
- **Store to Warehouse** - The transfer can be RMS initiated or SIM initiated that is it can be created in RMS or SIM. Source of transfer will be store whereas destination of transfer will be warehouse. Transfer will be shipped from source store in SIM and it will be received in destination warehouse in WMS by using appointment type ASN. Products included in this scenario: RMS, SIM and WMS.
- **Warehouse to Store** - The transfer can be RMS initiated or WMS initiated that is it can be created in RMS or WMS. Source of transfer will be warehouse whereas destination of transfer will be store. Transfer will be shipped from source warehouse in WMS using trailer and it will be received in destination store in SIM. Products included in this scenario: RMS, SIM and WMS.
- **Warehouse to Warehouse** - The transfer can be RMS initiated or WMS initiated that is it can be created in RMS or WMS. Source of transfer will be warehouse whereas destination of transfer will be warehouse. Transfer will be shipped from source warehouse in WMS using trailer and it will be received in destination warehouse in WMS by using appointment type ASN. Products included in this scenario: RMS and WMS.

### 3.2.5 Return to Vendor

Return to Vendor (RTV) is the scenario in which the stocks are returned back to the supplier from which it was received earlier. This return can be from a warehouse or store. One can return any amount of quantity he want to return or even the entire stock for an particular item.

### 3.2.6 Replenishment

Replenishment is an order for additional goods from a warehouse or supplier, in order to replenish depleted stock in a store or warehouse. Replenishment in RMS allows retailers to set up the automatic ordering of items, and RMS can monitor the inventory positions at locations throughout a retail enterprise, down to the item/location levels. Depending on the algorithm used, RMS replenishment can be configured to make recommendations, which can be manually added to a purchase order or transfer, or it can create purchase orders or transfers directly, depending on the level of automation desired.

## 3.3 Detailed Example Depicting the Flow and Validation Procedure

### 3.3.1 Scenario

**Store (A) to Store (B) Transfer initiated through Retail Merchandising System**

In this case the things that we are worried about the initial and final stocks at both the stores (A & B). The final stock for Stock A should be decreased by the quantity requested whereas the stock values for Store B should increase by the requested quantity. As this scenario includes Store hence OpenScript as well as JDeveloper are the software that will be needing for its automation.

The detailed step by step execution of Store to Store transfer is shown in Figure 3.2, it gives an overview about the tables that have to be checked after every step.

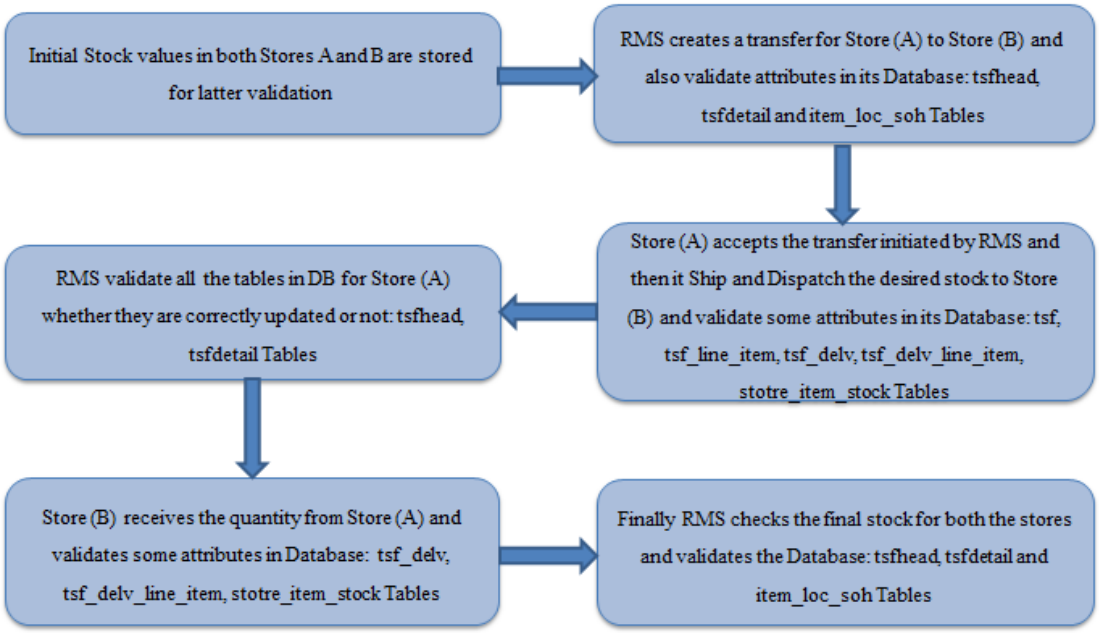



Figure 3.2 Store (A) to Store (B) Transfer initiated through RMS

As in this scenario OpenScript and JDeveloper are used for automation process, below is the detailed description of what is done at every step,

All the scripts that have been written for automation testing have a driver code which is mostly written in Openscript. The driver code has number of functions defined which are called at the required time. How different functions are called (in this scenario, three different functions) when required is shown in Figure 3.3. The order in which functions are called is,

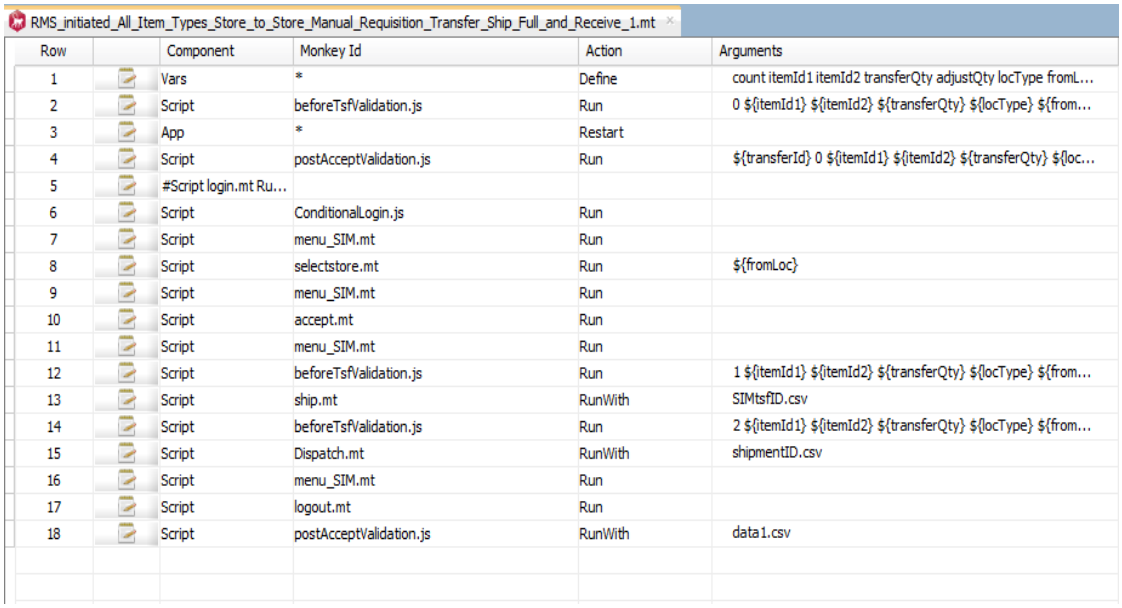
- **createTransfer() -** This function is responsible for creating Transfer for Store A to Store B of the desired quantity.
- **validatePostShipment() -** This function validates the value after transfer is shipped from Store A to Store B.
- **validatePostReceipt() -** This function validates the value after transfer is received at Store B.

```
public void run() throws Exception {

    try{


        Map<String,String> inputParamsMap = EIT_Util.prepareTestDataMap(getSettings().get("customParameters"));

        String testCaseFilePath = getScriptPackage().getRepositoryPath();
        //String propFilePath = testCaseFilePath.substring(0,testCaseFilePath.lastIndexOf("\\"))+"\\Environments\\RMS.properties";
        String propFilePath = testCaseFilePath+"\\Environments\\RMS.properties";
        Properties prop = EIT_Util.getProductProperties(propFilePath);

        if(null!=prop){
            inputParamsMap.put("rmsURL",prop.getProperty("RMS_URL"));
            inputParamsMap.put("rmsUser",prop.getProperty("RMS_Login_User"));
            inputParamsMap.put("rmsPassword",prop.getProperty("RMS_Login_Pwd"));
        }

        System.out.println(" ##### inputParamsMap from open script 728962_27252_RMS.SIM.New.All Item Types.Store to Store Intercompany Transfer ##### ..."+inputParamsMap);

        if(null== inputParamsMap.get("initialSohFromLoc") || "".equals(inputParamsMap.get("initialSohFromLoc")))
        {
            createTransfer(inputParamsMap,prop);

        }else if(null== inputParamsMap.get("received") || "".equals(inputParamsMap.get("received")))
        {
            validatePostShipment(inputParamsMap,prop);
        }
        else{
            validatePostReceipt(inputParamsMap,prop);
        }

    }
    catch(Exception e){
        System.out.println("Exception in open script Create PO for Regular Item....."+e);
    }
```

Figure 3.3 OpenScript code snapshot

The view of code from JDeveloper side which act as the driver code for the MonkeyTalk part for Dispatching is shown in Figure 3.4. It calls the required script at the desired time. Here it calls different main scripts in the order,

- **accept.mt** – This script is responsible for accepting the transfer from Store A to Store B at Store A. This is important because its RMS initiated. The initiation store first checks if it's having the stocks that will satisfy the ordered quantity or not. If condition is satisfied then only further execution is done else script fails.
- **ship.mt** – This script is responsible for shipping the container with the ordered quantity.
- **Dispatch.mt** – This script is used for dispatching the container to Store B location.
- **beforeTsfValidations.js, postAcceptValidation.js and ConditionalLogin.js** are used for validations in between process and for login into the application.



RMS_initiated_All_Item_Types_Store_to_Store_Manual_Requisition_Transfer_Ship_Full_and_Receive_1.mt

| Row | | Component | Monkey Id | Action | Arguments |
|---|---|---|---|---|---|
| 1 | | Vars | * | Define | count itemId1 itemId2 transferQty adjustQty locType fromL... |
| 2 | | Script | beforeTsfValidation.js | Run | 0 ${itemId1} ${itemId2} ${transferQty} ${locType} ${from... |
| 3 | | App | * | Restart | |
| 4 | | Script | postAcceptValidation.js | Run | ${transferId} 0 ${itemId1} ${itemId2} ${transferQty} ${loc... |
| 5 | | #Script login.mt Ru... | | | |
| 6 | | Script | ConditionalLogin.js | Run | |
| 7 | | Script | menu_SIM.mt | Run | |
| 8 | | Script | selectstore.mt | Run | ${fromLoc} |
| 9 | | Script | menu_SIM.mt | Run | |
| 10 | | Script | accept.mt | Run | |
| 11 | | Script | menu_SIM.mt | Run | |
| 12 | | Script | beforeTsfValidation.js | Run | 1 ${itemId1} ${itemId2} ${transferQty} ${locType} ${from... |
| 13 | | Script | ship.mt | RunWith | SIMtsfID.csv |
| 14 | | Script | beforeTsfValidation.js | Run | 2 ${itemId1} ${itemId2} ${transferQty} ${locType} ${from... |
| 15 | | Script | Dispatch.mt | RunWith | shipmentID.csv |
| 16 | | Script | menu_SIM.mt | Run | |
| 17 | | Script | logout.mt | Run | |
| 18 | | Script | postAcceptValidation.js | RunWith | data1.csv |

Figure 3.4 JDeveloper code snapshot for  Dispatching Quantity

The view of code from JDeveloper side which act as the driver code for the MonkeyTalk part for Receiving is shown in Figure 3.5. It calls the required script at the desired time. Here it calls different main scripts in the order,

- **receive.mt** – This script is responsible for receiving the ordered quantity at destination store B.
- **receivingValidation.js and ConditionalLogin.js** are used for validations after receiving process and for login into the application.

RMS_initiated_All_Item_Types_Store_to_Store_Manual_Requisition_Transfer_Ship_Full_and_Receive_2.mt

| Row | Component | Monkey Id | Action | Arguments |
|---|---|---|---|---|
| 1 | App | * | Restart | |
| 2 | Vars | * | Define | transferId count itemId1 transferQty adjQty locType fromL... |
| 3 | #Script login.mt Ru... | | | |
| 4 | Script | ConditionalLogin.js | Run | |
| 5 | Script | menu_SIM.mt | Run | |
| 6 | Script | selectstore.mt | Run | ${toLoc} |
| 7 | Script | menu_SIM.mt | Run | |
| 8 | Script | receive.mt | RunWith | data.csv |
| 9 | Script | menu_SIM.mt | Run | |
| 10 | Script | logout.mt | Run | |
| 11 | Script | receivingValidation.js | RunWith | data2.csv |

Figure 3.5 JDeveloper code snapshot for Receiving Quantity

### 3.3.2 Validations at Every Step

The attributes [12,13] that are being validated in different tables are as follow:

**Tsfhead table**

In this table the below listed attributes are validated after every step with the following values,

Table 3.1 Validation of attributes in tsfhead (RMS)

| Stages | Tsf_no | From_loc | To_loc | Status |
|---|---|---|---|---|
| Approved | Constant | Constant | Constant | **A** |
| Shipped | Constant | Constant | Constant | **S** |
| Dispatched | Constant | Constant | Constant | **D** |
| Received | Constant | Constant | Constant | **R** |

**Tsfdetail table**

In this table the below listed attributes are validated after every step with the following values,

Table 3.2 Validation of attributes in tsfdetail (RMS)

| **Stages** | **Tsf_no** | **Item** | **Tsf_qty** | **Ship_qty** | **Received_qty** |
|---|---|---|---|---|---|
| Approved | Constant | Constant | Constant | **0** | **0** |
| Shipped | Constant | Constant | Constant | **Quantity** | **0** |
| Dispatched | Constant | Constant | Constant | **Quantity** | **0** |
| Received | Constant | Constant | Constant | **0** | **Quantity** |

**Item_loc_soh tables**

In this table the below listed attributes are validated after every step with the following values,

On Store (A) side,

Table 3.3 Validation of attributes in item_loc_soh for Store A (RMS)

| **Stages** | **Item** | **Loc** | **Stock_on_ha-nd** | **In_transi-t_qty** | **Tsf_reserved_-qty** |
|---|---|---|---|---|---|
| Approved | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Shipped | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Dispatch-ed | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Received | Constant | Constant | **Previous - Quantity** | Previous | **Previous** |

On Store (B) side,

Table 3.4 Validation of attributes in item_loc_soh for Store B (RMS)

| **Stages** | **Item** | **Loc** | **Stock_on_h-and** | **In_transit_-qty** | **Tsf_reserve-d_qty** |
|---|---|---|---|---|---|
| Approved | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Shipped | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Dispatch-ed | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Received | Constant | Constant | **Previous + Quantity** | **Previous** | Previous |

**Tsf table**

In this table the below listed attributes are validated after every step with the following values,

Table 3.5 Validation of attributes in tsf (SIM)

| **Stages** | **External_id** | **Destination_id** | **Status** |
|---|---|---|---|
| Approved | Constant | Constant | **1** |
| Shipped | Constant | Constant | **2** |
| Dispatched | Constant | Constant | **7** |
| Received | Constant | Constant | **9** |

**Tsf_line_item Table**

In this table the below listed attributes are validated after every step with the following values,

Table 3.6 Validation of attributes in tsf_line_item (SIM)

| Stages | Tsf_no | Item | Quantity_requested | Quantity_received |
|---|---|---|---|---|
| Approved | Constant | Constant | **Quantity** | **0** |
| Shipped | Constant | Constant | **Quantity** | **Quantity** |
| Dispatched | Constant | Constant | **Quantity** | **Quantity** |
| Received | Constant | Constant | **Quantity** | **Quantity** |

**Store_item_stock tables**

In this table the below listed attributes are validated after every step with the following values,

On Store (A) side,

Table 3.7 Validation of attributes in item_loc_soh for Store A (SIM)

| Stages | Item_id | Store_id | Stock | In_transit_qty | Tsf_reserved_qty |
|---|---|---|---|---|---|
| Approved | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Shipped | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Dispatched | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Received | Constant | Constant | **Previous - Quantity** | Previous | **Previous** |

On Store (B) side,

Table 3.8 Validation of attributes in item_loc_soh for Store B (SIM)

| **Stages** | **Item_id** | **Store_id** | **Stock** | **In_transit_qty** | **Tsf_reserved_qty** |
|---|---|---|---|---|---|
| Approved | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Shipped | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Dispatched | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Received | Constant | Constant | **Previous + Quantity** | **Previous** | Previous |

If everything works as desired then these should be the validation values. As values are being validated after every step, thus if anything goes wrong one came to know immediately. The reason behind the un-wanted behavior are mainly be due to RIBs issues, application server side issues, Web Service issues. These issues can be easily seen through EIT Dashboard.

# Chapter 4

# Features and Implementation

## 4.1 Automation Portal and Controller

- One Click Automation – Instead of launching one-one script from a tool (OpenScript, MonkeyTalk or Synergy), launching it in a bunch easily by just selecting the scripts from a Portal.
- Constant Execution against live environment – Can be used for overnight execution against specific environment.
- Publish Execution Results email – Result of execution should be sent to the team as and when the execution completes by email.

### 4.1.1 Requirement for Portal

- To run OpenScript & Synergy on same machine – Previously the automation team was using two machines to run a script because the tools use different java for functioning. So using 2 machines was wastage of resource.
- On demand run for cloud and On Prem Env's – For cloud environment, DB access will not be there so need to add UI validations.
- On demand run for selected test case – Easy selection and launching of test script.
- Track the current execution – Can track the live execution of test scripts.
- Historic view of test case run results – Can access the old results and view it in a report mode as well chart mode.

### 4.1.2 Portal and Controller Design

The main structure of Portal can be seen from Figure 4.1. Its main components are Master Spreadsheet, Integration Hub, Property Files and Controller Program. Few characteristics of Portal can are listed below.

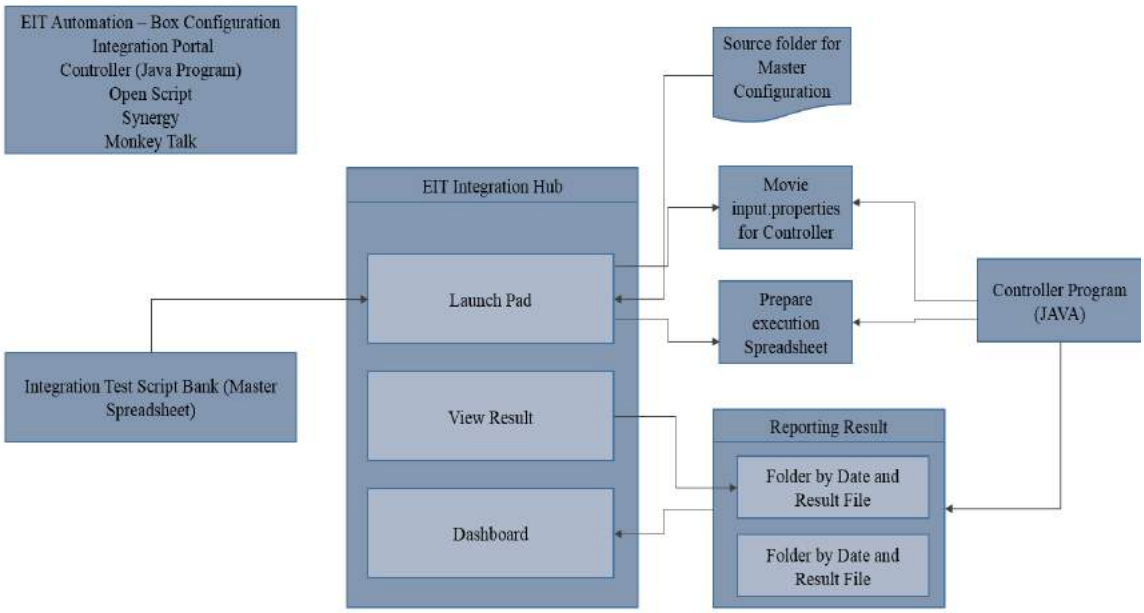
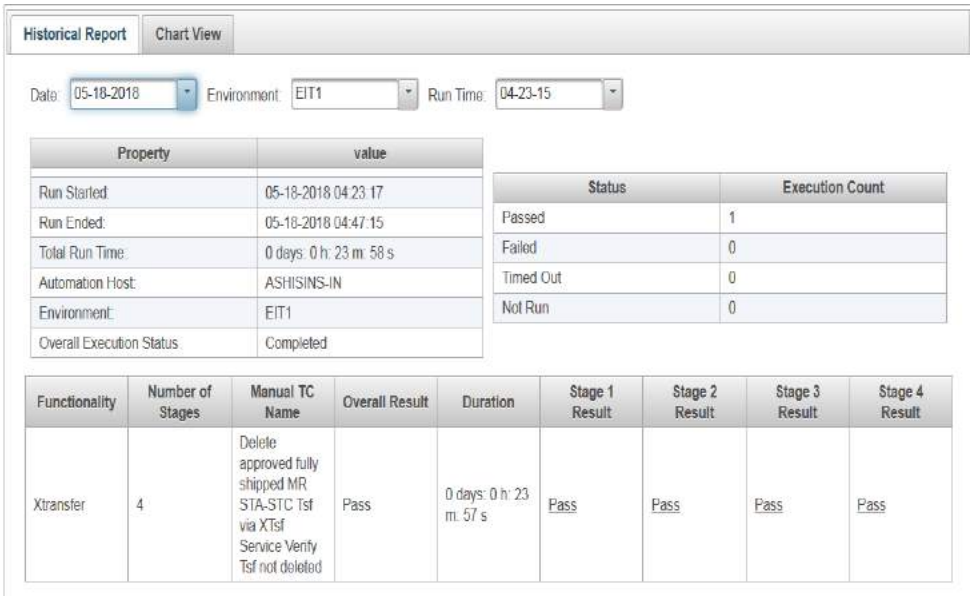



Figure 4.1 Portal and controller integration

- Read the file having test case name, data, path, etc. Read two excel file named "Execution_Commander.xlsx" and "Execution_Report.xlsx". Execution_Commander file is used to read the test case path and data whereas Execution_Report file will be having the result of execution. (Stage wise result, Overall result, total Runtime, total TC passed & failed as shown in Figure 4.2)
- Read the environment file against which the test case will be executed. Read the properties file move the product specific property file to OpenScript and Synergy Environments folder which will be used while execution and the same can be seen from Figure 4.3.
- Run the OpenScript and Synergy on same machine. Synergy uses the JAVA_TOOL_OPTIONS env. Variable which cannot be used for OpenScript so before starting the OpenScript, controller will remove this env. Variable.
- Parse the Result xml file of OpenScript and synergy to get the result. To get the final result, controller will parse the tags of result xml file and put the result (Pass/Fail) in Execution_Report. Save the result of stages in file.
- Publish Email Result. After the completion of execution, the result will be sent to the team.



Historical Report | Chart View

Date: 05-18-2018 Environment: EIT1 Run Time: 04-23-15

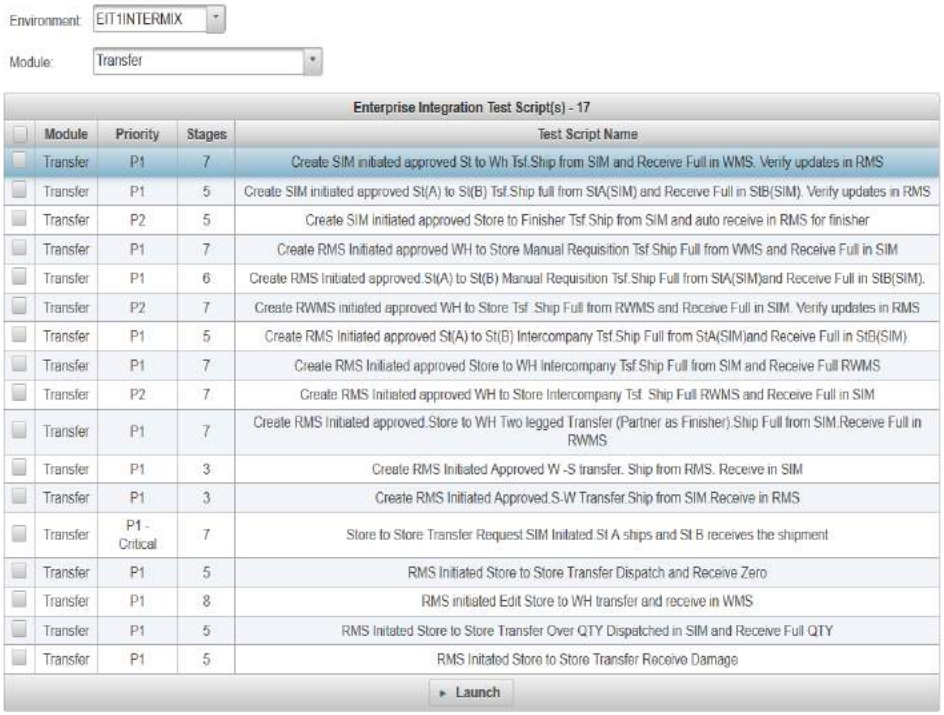
| Property | value |
|---|---|
| Run Started: | 05-18-2018 04:23:17 |
| Run Ended: | 05-18-2018 04:47:15 |
| Total Run Time: | 0 days: 0 h: 23 m: 58 s |
| Automation Host: | ASHISINS-IN |
| Environment: | EIT1 |
| Overall Execution Status: | Completed |

| Status | Execution Count |
|---|---|
| Passed | 1 |
| Failed | 0 |
| Timed Out | 0 |
| Not Run | 0 |

| Functionality | Number of Stages | Manual TC Name | Overall Result | Duration | Stage 1 Result | Stage 2 Result | Stage 3 Result | Stage 4 Result |
|---|---|---|---|---|---|---|---|---|
| Xtransfer | 4 | Delete approved fully shipped MR STA-STC Tsf via XTsf Service Verify Tsf not deleted | Pass | 0 days: 0 h: 23 m: 57 s | Pass | Pass | Pass | Pass |

Figure 4.2 Detailed description of result after execution

- In case of failure of initial stages, block the rest of stages and fail that test case.



Environment: EIT1INTERMIX

Module: Transfer

Enterprise Integration Test Script(s) - 17

| Module | Priority | Stages | Test Script Name |
|---|---|---|---|
| Transfer | P1 | 7 | Create SIM initiated approved St to Wh Tsf.Ship from SIM and Receive Full in WMS. Verify updates in RMS |
| Transfer | P1 | 5 | Create SIM initiated approved St(A) to St(B) Tsf.Ship full from StA(SIM) and Receive Full in StB(SIM). Verify updates in RMS |
| Transfer | P2 | 5 | Create SIM initiated approved Store to Finisher Tsf.Ship from SIM and auto receive in RMS for finisher |
| Transfer | P1 | 7 | Create RMS Initiated approved WH to Store Manual Requisition Tsf.Ship Full from WMS and Receive Full in SIM |
| Transfer | P1 | 6 | Create RMS Initiated approved.St(A) to St(B) Manual Requisition Tsf.Ship Full from StA(SIM)and Receive Full in StB(SIM). |
| Transfer | P2 | 7 | Create RWMS initiated approved WH to Store Tsf .Ship Full from RWMS and Receive Full in SIM. Verify updates in RMS |
| Transfer | P1 | 5 | Create RMS Initiated approved St(A) to St(B) Intercompany Tsf.Ship Full from StA(SIM)and Receive Full in StB(SIM) |
| Transfer | P1 | 7 | Create RMS Initiated approved Store to WH Intercompany Tsf.Ship Full from SIM and Receive Full RWMS |
| Transfer | P2 | 7 | Create RMS Initiated approved WH to Store Intercompany Tsf .Ship Full RWMS and Receive Full in SIM |
| Transfer | P1 | 7 | Create RMS Initiated approved.Store to WH Two legged Transfer (Partner as Finisher).Ship Full from SIM.Receive Full in RWMS |
| Transfer | P1 | 3 | Create RMS Initiated Approved W -S transfer. Ship from RMS. Receive in SIM |
| Transfer | P1 | 3 | Create RMS Initiated Approved.S-W Transfer.Ship from SIM.Receive in RMS |
| Transfer | P1 - Critical | 7 | Store to Store Transfer Request.SIM Initated.St A ships and St B receives the shipment |
| Transfer | P1 | 5 | RMS Initiated Store to Store Transfer Dispatch and Receive Zero |
| Transfer | P1 | 8 | RMS initiated Edit Store to WH transfer and receive in WMS |
| Transfer | P1 | 5 | RMS Initated Store to Store Transfer Over QTY Dispatched in SIM and Receive Full QTY |
| Transfer | P1 | 5 | RMS Initated Store to Store Transfer Receive Damage |

Launch

Figure 4.3 Requirements for executing a test case through automation

## 4.2 Application Status Check

Application status check task is to facilitate in the automation of nightly scheduled run on daily basis only if the applications that are in use have their RIBs up and working properly. For now we were kicking off the nightly run in different production servers without checking the status of RIBs, due to which the failure counts were increasing. To compensate this, application status check utility is developed.

### 4.2.1 Table Requirements

Table nam "Automation_config_check" was designed with the following fields,

- **Environment –** It shows the running environment of test scripts like EIT1, EIT2, QA7, QA8 etc.
- **HOST –** It shows the name of production server from where the run in kick off.
- **RUN_DATE –** This field stores the date of run of configuration scripts.
- **RMS, RWMS, SIM, RPM, ALLOCATION, RI, RDE, RESA, RMS_WEB, IGS_WEB, RIB, REIM –** These are fields for different application which stores UP/DOWN according to the status of their RIBs.

The snapshot of Automation_config_check table with different fields is show in Figure 4.4.

| | ENVIRONMENT | HOST | MODULE | RMS | RWMS | SIM | RPM | RUN_DATE | ALLOCATION | RI | RDE | RESA | RMS_WEB | IGS_WEB | SIM_WEB | RIB | REIM | NIGHTLY_BATCH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 86 | EIT2 | Blr2223131 | (null) | DOWN | (null) | (null) | DOWN | 11-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 85 | EIT1 | EIT_Auto_Prod-1 | (null) | UP | UP | UP | UP | 11-NOV-17 | UP | UP | UP | UP | UP | UP | (null) | UP | UP | UP |
| 84 | EIT2 | Blr2223131 | (null) | DOWN | (null) | (null) | DOWN | 10-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 83 | EIT1 | EIT_Auto_Prod-1 | (null) | UP | UP | UP | UP | 10-NOV-17 | DOWN | UP | UP | UP | UP | UP | (null) | UP | UP | UP |
| 82 | EIT2 | Blr2223131 | (null) | DOWN | (null) | (null) | DOWN | 09-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 81 | EIT1 | Blr2223131 | (null) | UP | UP | UP | UP | 09-NOV-17 | DOWN | UP | UP | UP | UP | UP | (null) | UP | UP | UP |
| 80 | EIT2 | Blr2223131 | (null) | DOWN | (null) | (null) | DOWN | 08-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 79 | EIT1 | EIT_Auto_Prod-1 | (null) | UP | UP | UP | UP | 08-NOV-17 | DOWN | UP | UP | UP | UP | UP | (null) | UP | UP | (null) |
| 78 | EIT2 | Blr2223131 | null | DOWN | (null) | (null) | DOWN | 07-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 77 | EIT1 | Blr2223131 | null | UP | UP | UP | UP | 07-NOV-17 | UP | UP | UP | UP | UP | UP | (null) | UP | UP | UP |

Figure 4.4 Data in Automation_config_check table

### 4.2.2 Constraints for Table

"Automation_config_check" table gets updated only if RIB.properties is having "Integration_flg" as "Yes". How to set integration_flag variable value can be seen in Figure 4.5. Only one entry is made for particular environment in a day and multiple run would simply update their status.

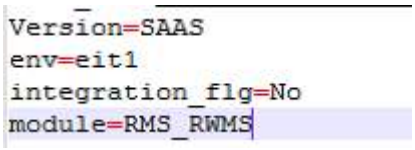



Figure 4.5 Some fields from RIB.properties file

### 4.2.3 Requirements of Bat File

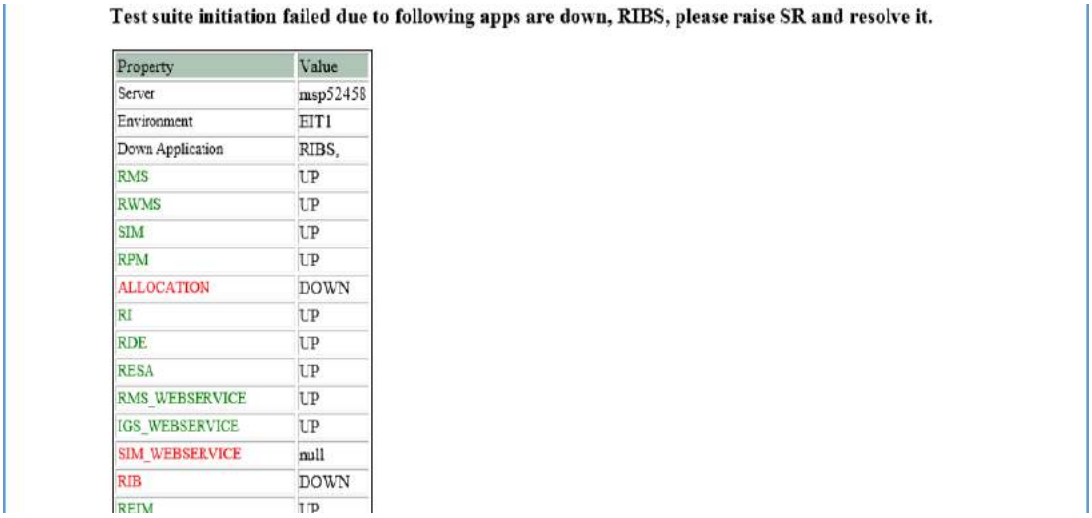
There is a "run.bat" file which is the first program that executes when it comes to automation. The main task of bat file is to facilitate the environment for automation. The very first thing it do is moves the files from gold folder to master folder and then it calls "checkNightlyRun.java" which checks the status of applications. If the status of dependent applications is UP then this bat file calls "Automation_Executor.java" and the run will kick off or else it just sends a mail showing the RIBs are down so nightly run is not initiated.

### 4.2.4 Other Requirements and Constraints

The very first thing that is needed to run status check task is that all the configuration scripts must be placed in the desired production server and the gold folder is having all the required files in it and also the automation executor is place at its desired location. Either update or insert query is executed in each configuration script corresponding to the status of the application in "Automation_config_check" table. "integration_flg" is the main factor that is responsible for application status check task, if its value is set to Y, then only the values will be entered in database and the scheduled check run will start else the nightly run is kicked off directly.

### 4.2.5 Workflow

At the initial stage, before running the automation_executor.java from the production server, the configuration scripts placed in that production server is executed. This configuration script checks the status of dependent applications if the “integration-flg” is set to ‘Yes’. Once the configuration script is executed then only the “checkNightlyRun.java” file is executed. This file update the values of status as UP/DOWN in “Automation_config_check” table. Once the table value is populated with UP the nightly run is kicked off else the mail is sent with details of applications whose status are DOWN that RIBs are down and the nightly run cannot be started. Figure 4.6 shows the mail content that is generated with the property and their values and sent when the applications are down.



**Test suite initiation failed due to following apps are down, RIBS, please raise SR and resolve it.**

| Property | Value |
|---|---|
| Server | msp52458 |
| Environment | EIT1 |
| Down Application | RIBS, |
| RMS | UP |
| RWMS | UP |
| SIM | UP |
| RPM | UP |
| ALLOCATION | DOWN |
| RI | UP |
| RDE | UP |
| RESA | UP |
| RMS_WEBSERVICE | UP |
| IGS_WEBSERVICE | UP |
| SIM_WEBSERVICE | null |
| RIB | DOWN |
| REIM | UP |

Figure 4.6 Mail, if some applications are down

## 4.3 EIT Tools and Utility Portal

**Administration:** Administration block contains various utilities like Add User, Add New Connection and Database Connection Check.

**EIT Dashboard:** Graphical representation of analysis of analysis of results and foundation data of various batches is shown in EIT Dashboard.

**EIT Reports:** Tools for generating system options, reports for messages covered, foundation data variables and its comparison report across environments are present in EIT Report.

**EIT Tools:** The functionalities like Configuration management, Data verification, POSU upload generation and Transaction validation are present in EIT Tools.

Figure 4.7 show the UI of the EIT Tools and Utilities Portal.

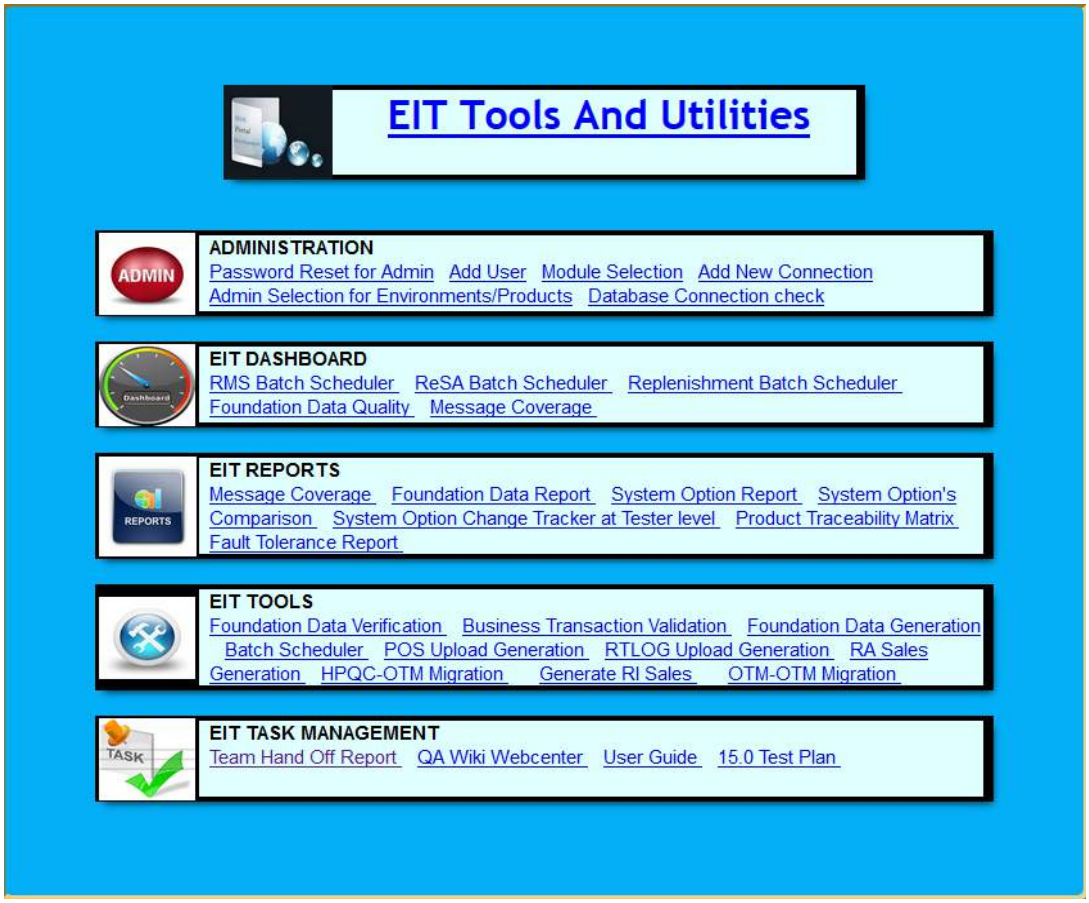



Figure 4.7 EIT Tools and Utilities Portal

### 4.3.1 POS Upload Generation

This utility is used for generating the pre required data for executing the automation scripts for some scenarios like POS Files and X-Store. POS Files and X-Store are used to simulate day to day interaction between the vendor (Shopkeeper) and the customer. The operations under this can be,

- **Sales** – It simulates normal sale of same type of item from the shop.
- **Return** – It simulates normal return of an item to the shop.
- **Exchange** – It simulates normal exchange of an item in the shop i.e. and customer had brought and item in past but now he wants some different item in place of already brought item.
- **Layaway –** It simulates the case where a customer had paid the half amount of the total bill and thus the vendor (shopkeeper) reserves the amount for the particular customer i.e. now logically that quantity is subtracted from the total stock but physically it still exists.
- **Mixed Transaction –** It is combination of Sales and Return in the single transaction.

All of the above files are in .dat format. A simple Sales .dat file example is given in Figure 4.8,

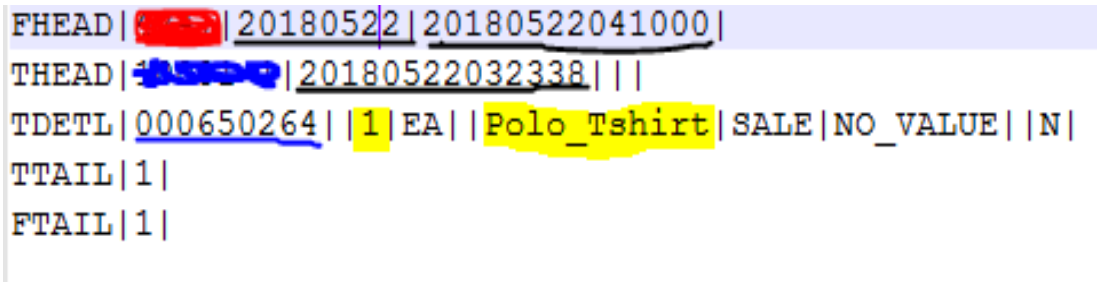



Figure 4.8 Snapshot of simple Sales POS File

The POS File usually contains FHEAD (File Head), THEAD (Table Head), TDETL (Table detail), TTAIL (Table Tail) and FTAIL (File Tail). FHEAD has details about the store for which the transaction is created and also the timestamp. THEAD contains a unique transaction id with which every single transaction can be distinguished. TDETL is the main field in .dat file as it contain the item, quantity as well as the description for the item. TTAIL is the count of how many TDETL are there in a single transaction. There can be many TDETL in a single .dat file.

These files are generated from the utility portal by following the steps in Figure 4.9 and Figure 4.10,


Figure 4.9 POS File Generation

All the required fields are entered for which a POS File has to be created and then it can be simply downloaded from the server.


Figure 4.10 POS File for which .dat file will be created

## 4.4 Continuous Delivery Dashboard

Continuous Delivery Dashboard is used for monitoring and analyzing automation run and its result. Script run results of all the servers can be visualized which helps in determining the failure reason and analyzing the results generated. It reduces the time involved in manual analysis of reports.

### 4.4.1 Dashboard Features

Developed using Express js framework for nodejs for analyzing automation scripts result. Continuous Delivery Dashboard is hosted in Tomcat Server. Dashboard's features are as follow,

- **Dashboard (velocity)** - This page consists of an aggregate of last three months report with Passed, Failed and Total count of the scripts. Data is visualized in bar chart, pie chart and in table format. The snapshot of the report for month of March, April and May with bar graph and pie chart is shown in Figure 4.11.


Figure 4.11 Pie chart representation of Passed, Failed and Total count of scripts

- **Execution Trend Report** - This Page consist of daily run, weekly run and monthly run report. Bar graph and pie chart is generated based on daily, weekly, monthly

number of pass and fail in all servers. The snapshot of the execution trend for month of May with bar graph and pie chart is shown in Figure 4.13.

- **Script Delivery Report** - This Page is to keep track of Script Development Progress. Line graph is generated based on number of planned Scripts versus actual no of Scripts developed. Data is visualized in Line graph and table format and a snapshot can be seen from Figure 4.12.

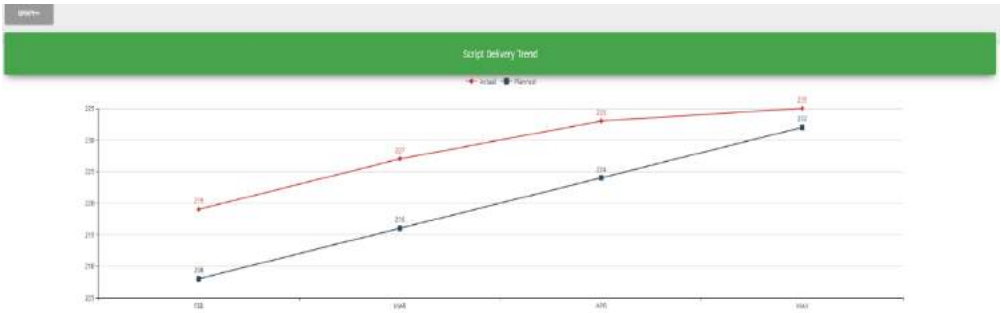



Figure 4.12 Graphical representation of script delivery report


Figure 4.13 Pie chart representation of execution trend

- **Average Execution** - This Page contains all duration of runs in hours and minutes with no of scripts, no of iteration for each server for monthly basis.

# Chapter 5

# Result and Analysis

The observations and findings while working on this project mainly depict why one technology is preferable than the other one or why there was a shift from one technology to another. Also, it shows how one can constantly make changes in existing system to make it more reliable and efficient.

## 5.1 Result of Automation Script

The result can be easily seen from the Automation portal (internal utility which provide one click automation). Multiple scripts can be selected and executed at a time. If single script is selected then the result will appear in the following way with information about Start time, End time, Status of script, Auto Prod in which it was executed, Passed/Failed counts and much more. The Figure 5.1 shows the result for the scenario that we had discussed in section 3.3.1.

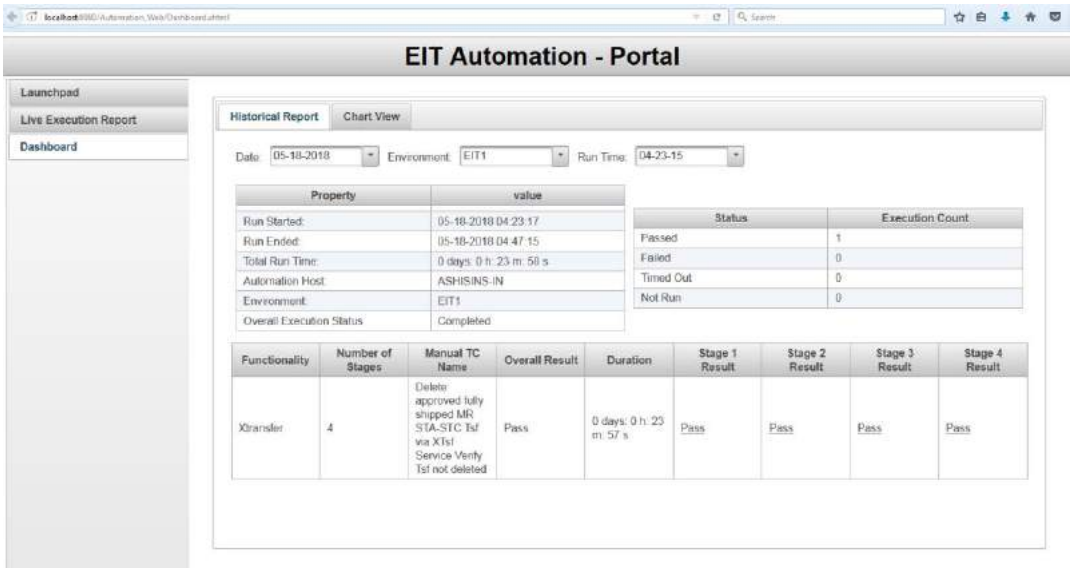



Figure 5.1 Result of the executed script

A detailed result can also be checked by clicking on the Stage Result hyperlink (Pass/Fail). This will show line wise execution result which is very helpful in debugging.

Through the detailed result we can easily know the cause behind the failuer of the scripts. The main reason is found after following the trace of failure i.e. if script is failed in validation part or somewhere while performing desired task then we try to find whats the real reason due to which it got failed as shown in Figure 5.2. There can be many reasons behind this, like if the RIBs are down, Application server itself is down or the desired operation was not performed due to some reasons and thus the validations differ.

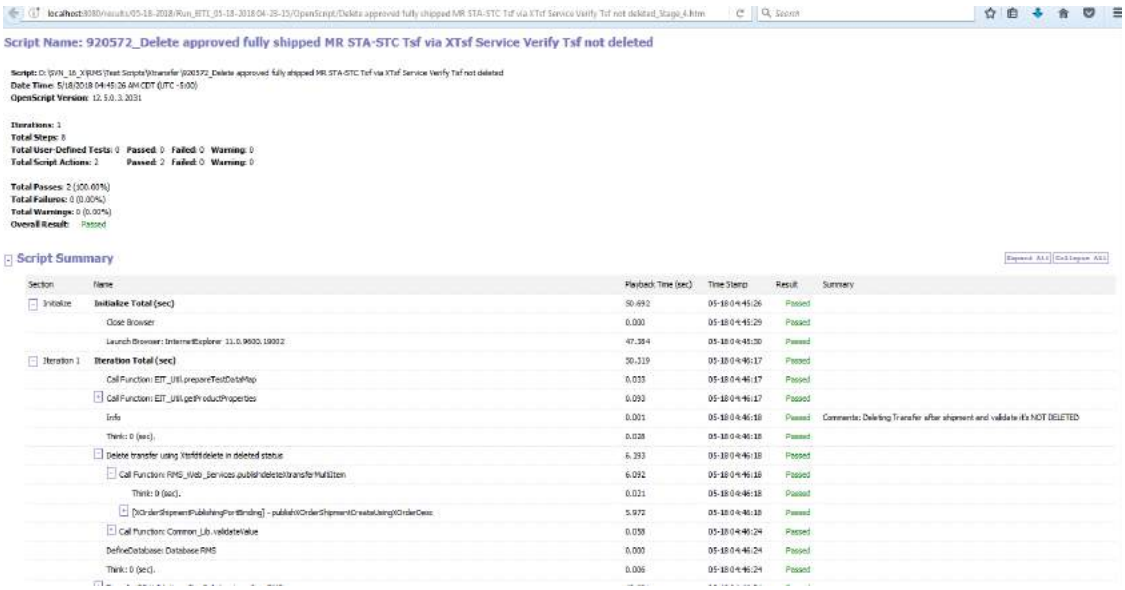



Figure 5.2 Detailed result of executed script

## 5.2 Reasons behind Failure of Scripts

There are several reasons behind the failure of an automation script and most commonly occurring reasons are listed below,

### 5.2.1 Web Page Refreshing

As we work on Cloud environment so for better security the application is defined with several timestamps after which it gets refreshed so as to prevent certain threats. This thing many a times causes an automation script to fail.

### 5.2.2 RIBs

Retail Integration Bus is one of the most commonly occurring cause for scripts failure because a total count of 200+ scripts are running that are distributed into five different Production servers. Some scripts need RIBs to be down and some need the RIBs to be up and working. If at an instance two different scripts on different production server are running who require different state of RIB then one scripts will definitely fail.

### 5.2.3 Server Side Issue

Sometime the server is not responding properly or the Publishers and Subscribers are slow i.e. the data is taking time to flow through RIBs. Due to this also the script gets failed.

### 5.2.4 Application Side Issue (UI)

Sometimes after deployment (new build release) the scripts gets failed. With new advancements in the application some new features are added and old features are modified but the scripts are written according to the older version. Hence they gets failed.

### 5.2.5 Scripting Error

This is also one of the main reason behind the failure. While scripting, if proper testing of script with different scenarios is not done then there might be chance that script is having error (while writing script). Sometime only one condition is tested and it is passed onto the production but his should not be the case.

## 5.3 Contribution of each cause in Failure

When it comes to define the percentage of failure due to one particular cause then one should limit the causes so that a better understanding can be made. In our case, we have limited the causes to the main five reasons (discussed above) behind the failure of scripts. Table 5.1 shows the stats of failure reasons with their respective contribution towards total failure in scripts failure.

Table 5.1 Reasons with their contribution in scripts failure

| Reasons | Total Percentage | Contribution in Failure |
|---|---|---|
| Web Page Refreshing | 100 | 25 % |
| RIBs | 100 | 12 % |
| Server Side Issues | 100 | 24 % |
| Application Side Issues | 100 | 4 % |
| Scripting Error | 100 | 35 % |

These percentage are carefully calculated from the nightly run status by using the following formula,

*Contribution in Failure = Failures caused by particular issue / Total failure count*

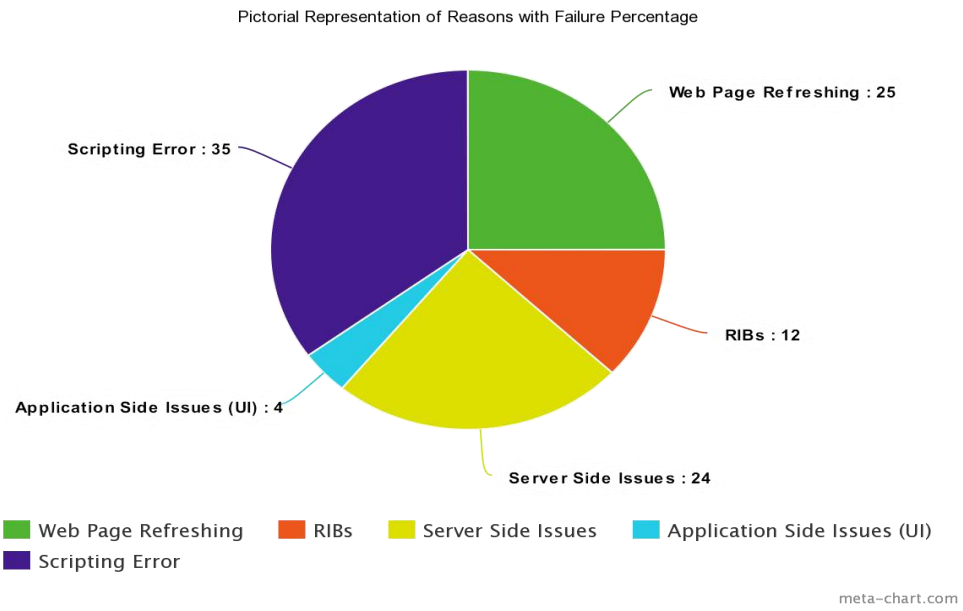



Figure 5.3 Pictorial Representation of reasons with failure percentage

The pictorial representation of failure reasons with their respective contribution towards total failure in scripts failure is shown in Figure 5.3.

## 5.4 Precautions for Reducing Number of Failures

- **Web Page Refreshing** – A proper amount of think time is added in the scripts which prevents the script from failure. As the page is refreshed after certain interval of time so we can add think time accordingly. But more think time leads to long run. Thus it should be added only where ever needed.
- **RIBs –** A proper check is done on all the production servers and then the sequence of execution of scripts in each of the production server is set. This has reduced the RIBs failure to a great extent.
- **Server Side Issues –** Usually this issue arises only when either of the Publishers or Subscriber is down or slow. We can add appropriate think time before moving to another application. Suppose if we are using RMS and SIM then before switching to another application we can add think time to overcome this issues. Sometimes Publishers and Subscribers are forcefully kept down so as to upgrade the application. In this case we can't do anything till the upgradation isn't complete.
- **Application Side Issue (UI) –** This issue logically doesn't have any preventive steps that we can opt for. In this case we must modify existing scripts according to the latest version of application and also keep in mind that previous functionalities should be there.
- **Scripting Error –** These types of error are human error which can only be reduced at the time of scripting.

# Chapter 6

# Conclusion and Future Work

After complete knowledge of the products and tools I worked on the testing of integrated Oracle products. At the later stage, I had to write Android app Testing Scripts using MonkeyTalk and OpenScript. The scripts were successfully tested and sent for production run on nightly basis.

## 6.1 Conclusion

Everyone is worried about the quality of product they are buying and this project helps in attaining a certain level of product quality. Through testing we can know what the limitations of the product are and how they can be overcome. As all the testing is done through automation thus a lot of time is reduced and also the chances of human error. The most occurring reasons that were responsible for degrading product quality are, web page refreshing, RIBs, server side issues, application side issues and scripting error. I would also like to add that automation is a never ending process because with upgradation of application automation might need to be changed.

## 6.2 Future Scope

Machine Learning and Data Mining techniques will be used in future to get better and appropriate reasons for script failure so that a much better software quality can be achieved. EIT Automation Portal will also be enhanced based on the requirement and also for getting deeper into the results of automation scripts. More scenarios of retail world will be covered through automation. All the earlier written scripts will also be kept in mind for consistency in scripts result. Most importantly, regular upgradation of scripts with upgradation of product is necessary else the script failure count will be increased.

# References


[1] Oracle Retail. Oracle functional testing. 2010.

[2] Tanupriya Choudhury Sai Sabitha Jai Gaur, Akshita Goyal. “A walk through of software testing techniques”. *International Conference System Modeling & Advancement in Research Trends (SMART)*, 2016. DOI: 10.1109/SYSMART.2016.7894499

[3] Prasad, Dr. K.V.K.K. “ISTQB Certification Study Guide”, *Wiley*, 2008.

[4] Nagwa Badr Passant Kandil, Sherin Moussa. “A methodology for regression testing reduction and prioritization of agile releases”. *5th International Conference on Information & Communication Technology and Accessibility (ICTA)*, 2015. DOI: 10.1109/ICTA.2015.7426903

[5] Black, Rex. “Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing”. *Wiley*, August 2009.

[6] Bernie Gauf Elfriede Dustin, Thom Garrett. “Implementing automated software testing”. *IEEE International Computer Software and Applications Conference*, 2009.

[7] Kolawa, Adam; Huizinga, Dorota. “Automated Defect Prevention: Best Practices in Software Management”. *Wiley-IEEE Computer Society Press*, 2007.

[8] Jingfan Tang. “Towards Automation in Software Test Life Cycle Based on Multi-Agent”. *IEEE*, 2010.

[9] Binder, Robert V. “Testing Object-Oriented Systems: Objects, Patterns, and Tools”. *Addison-Wesley Professional*, 2000. ISBN 978-0321700674.

[10] Claus Klammer and Rudolf Ramler. “A Journey from Manual Testing to Automated Test Generation in an Industry Project”. *IEEE International Conference on Software Quality, Reliability and Security*, 2017.

[11] Manjit Kaur, Raj Kumari, “Comparative Study of Automated Testing Tools: TestComplete and QuickTest Pro”. *International Journal of Computer Applications*, 2011.

[12] Oracle Retail. Oracle retail merchandising system. 2012.

[13] Oracle Retail. Oracle store inventory management. 2013.

[14] Oracle Retail. Oracle warehouse management system. 2013.

[15] Oracle Retail. Oracle retail invoice matching. 2013.

[16] Oracle Retail. Oracle retail price management. 2014.

[17] Oracle Retail. Oracle allocation. 2015.

[18] Oracle Retail. Oracle retail integration bus. 2012.

[19] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. "The Oracle Problem in Software Testing". *IEEE transactions on software engineering, vol. 41, no. 5*, may 2015.

[20] Harpreet Kaur, Dr.Gagan Gupta. "Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete". *Int. Journal of Engineering Research and Applications*, 2013. ISSN : 2248-9622.

[21] Jagdish Singh, Monika Sharma. "A Comprehensive Review of Web-based Automation Testing Tools". *International Journal of Innovative Research in Computer and Communication Engineering*, 2015.

[22] K. Valliammai, Dr. P. Sujatha. "Analysis of Efficiency of Automated Software Testing Methods: Direction of Research". *International Journal of Science and Research*.

[23] Vishawjyoti and Sachin Sharma. "Study and analysis of automation testing techniques". *Journal of Global Research in Computer Science*, 2012.

# Acknowledgement

This research work cannot be marked as completed without acknowledging the ones who supported and guided me for the successful completion of this task.

It gives me enormous pleasure in showing courtesy and deep gratitude to **Mr. Puneet Saxena**, Senior Manager, Oracle RGBU, Minneapolis MN, US and **Mr. Advait Bhatt**, Quality Assurance Engineer, Oracle RGBU, Bangalore, India for their valuable guidance and mentorship that helped me to overcome every challenge I faced as I moved on in this work.

I wish to show my deep gratitude to **Shatrughan Modi**, Lecturer, Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala for his worthy guidance and continuous assistance throughout my work. He has always created a motivational and supported environment for me. His intelligence has always prompted me in doing more and more and this will surely give me advantage in future life.

I am grateful to **Dr. Maninder Singh**, Hon'ble Head of Computer Science and Engineering Department, Thapar Institute of Engineering and Technology, Patiala for his kind support and providing basic access to technologies and the desired workspace.

I would also like to thank the institution, all faculty members of Computer Engineering Department, Thapar Institute of Engineering and Technology, Patiala for their special attention and suggestions towards this work.

**Ashish**

**801632004**

# Abstract

It is vital in any industry especially in IT to provide the right useful product to the desired customer with all the quality checks i.e. best quality one can ensure and that too at the right time. If we try to check the quality manually then a lot of time will be consumed as various input combinations has to be checked and it's also difficult to find all the defects because of human error. Moreover, it is costly, in term of resources, to manually analyze all results. These limitations were the main motivation behind this work so that we can overcome the limitations of manual testing on enterprise products using automation.

If we talk about manual testing, any change in the hardware configuration or a bit of change in source code leads to repetition of entire process. But in case of automation we have already pre-defined steps written through code that have to be followed and it is also very helpful in analyzing the results. Through automation we also gets a report for every run with all the details that can be used further for analysis purpose. One can also find the reason of failure of script through result that is generated through automation.

Enterprise Integration Testing (EIT) Tools and Utilities is what we came as a solution for automation. The major functionalities that comes under this are, EIT Tools, EIT Task Management, EIT Dashboard and EIT Reports with multiple utilities under each of them.

After using EIT Tools for regression testing, we came to know the most frequent reasons which were degrading the software quality and once these factors are highlighted then the preventive measures can be taken to improve the software quality.

# Table of Content

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 General Introduction to the Topic

It is vital in any industry especially in IT to provide the right useful product to the desired customer with all the quality checks i.e. best quality one can ensure and that too at the right time. If we try to check the quality manually then a lot of time will be consumed as various input combinations has to be checked and it's also difficult to find all the defects because of human error [1]. Moreover, it is costly, in term of resources, to manually analyze all results. These limitations were the main motivation behind this work so that we can overcome the limitations of manual testing on enterprise products using automation.

If we talk about manual testing, any change in the hardware configuration or a bit of change in source code leads to repetition of entire process. But in case of automation we have already pre-defined steps written through code that have to be followed and it is also very helpful in analyzing the results. Through automation we also gets a report for every run with all the details that can be used further for analysis purpose. One can also find the reason of failure of script through result that is generated through automation.

## 1.2 Need of Testing

The best quality of a product can be ensured through testing. Testing is the most crucial phase of development cycle. It helps in finding out more relevant information related to the quality of product and how it can be improved. It also checks for the flow with which the product should work. In all, testing can be used for giving a better vision to products.

Bugs finding, error findings and defect findings are the main objectives of testing which are not generally taken care in the development cycle. At development time, these are not desirable [19]. Testing techniques available for testing depends on the nature of testing like Integration testing, Quality Acceptance testing and Unit testing are some of them.

## 1.3 Different Types of Testing

Different types in which Software Testing can be classified are as follows [2],

- Security Testing
- Functional Testing
- Performance Testing
- Regression Testing
- User Acceptance Testing

Among the above mentioned testing methods, Regression Testing is most laborious, time consuming and error prone testing type as in it, whenever the system undergoes any kind of change then complete execution of all set of test cases are done. Hence, a term is introduced called as Test Automation which eases the task of Regression Testing.

### 1.3.1 Functional Testing

Functional testing is a type of black-box testing and a quality assurance (QA) process [3]. All the test cases are based on the specifications of the software component that is under test. Functions are generally tested by feeding them with input and then examining the output. Internal program structure is rarely considered in it. What the system does is checked in functional testing.

### 1.3.2 Regression Testing

Regression testing is major part of the integration testing. Whenever there is a change in some part of the system, then entire system undergoes testing to find any chances of failure. Regression testing comes out to be costly in terms of money and time if, complex application are considered.

The main goal of regression testing [4] is to make sure that with introduction of new changes the previous functionalities doesn't stop working in appropriate manner. It checks if the software quality is compromised with new changes or not. Test cases written for previous version of software can be for new version also. We must keep in mind that number of test cases should never grow with software.

Test suite selection techniques try to reduce the cost of testing by running a subset of the tests, such as those that execute the modified source code, in order to ensure that the updated program still operates correctly.

### 1.3.3 Performance Testing

Performance testing in software engineering is in general, a testing practice that is used to determine how system performs in terms of stability and responsiveness under a particular condition. It is also used to measure, verify, validate or investigate other quality attributes of the system, such as reliability, resource usage and scalability.

### 1.3.4 User Acceptance Testing

User acceptance testing is a test conducted to determine if the specification or contract are met with the told specifications. For acceptance testing we can perform performance test, chemical tests or physical tests based on the requirement. Black-box testing performed on a system prior to its delivery in systems engineering (for example: lots of manufactured mechanical parts, batches of chemical products, or a piece of software) [5].

### 1.3.5 Security Testing

Security testing is a way to identify flaws in the security mechanisms of the information system that is used to protect data and maintain functionality. As there are logical limitations in security testing thus passing security testing does not indicate that there are no flaws in the system but adequately satisfies the security requirements.

## 1.4 Motivation

Arrangement of tools, ideas and assumptions to provide some core functionalities was the main goal of test automation. Error recovering, monitoring, logging, functioning and reporting are the functionalities that it will have. There are different approaches to automation that use different methodology [6]. Automation mainly assign same job to computers and thus reduces the job of test engineers. If regression testing is to be done, then a person (manually) takes more time to execute test cases as he has to validate all the

attributes manually which takes a lot of time. The time taken in validations can be reduced if it's done through automation and more test cases can be executed on a particular day as described in Figure 1.1.

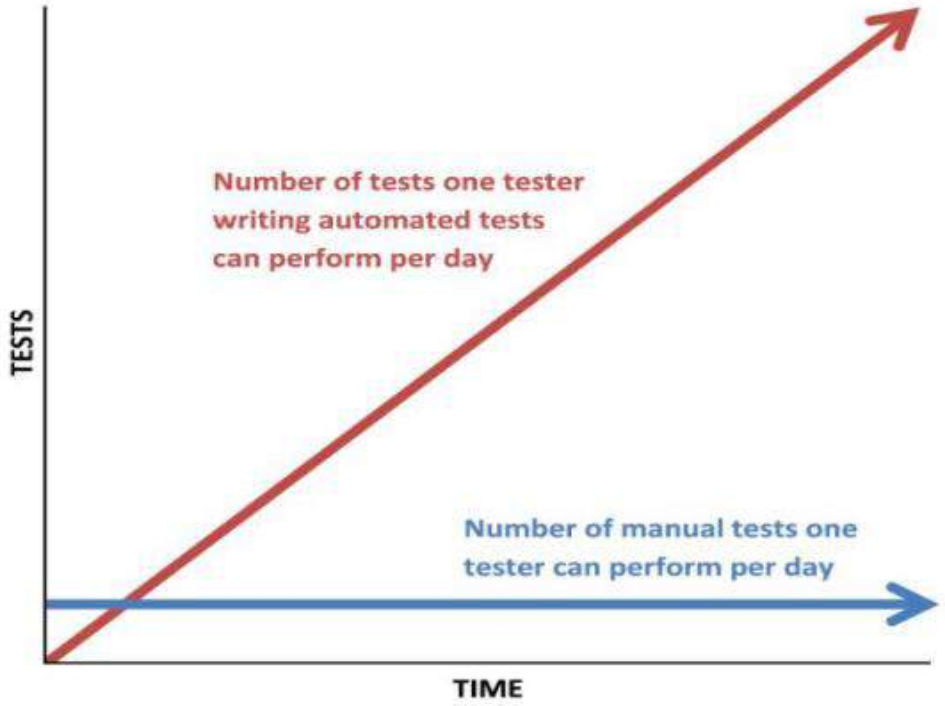



Figure 1.1 Comparison between Manual and Automation Testing

Database validations, load performance and compliance of requirements are what, that are included in automating manual testing process. New capabilities and features are also added with the growing demand of support in testing. Regression testing is preferable to find bug in the code and reduce risk in fastest way with less effort.

## 1.5 Problem Definition

Design, Development and Implementation of Automation Tool and Utilities which are used by Enterprise Integration Testing team for reducing manual efforts. As a total of 230+ scenarios are there that needed be tested regressively thus time plays a crucial role here. Manual Testing was not giving the desired result thus there was a need for introduction of Automated Testing that can regressively test 230+ scenarios efficiently.

## 1.6 Objectives

The main goal of this project is to automate functionalities which are regularly used in test execution in order to,

- Increasing productivity of manual testing process.
- Reduce cycle time of testing by automating functionalities and reducing efforts.
- Developing an easy to understand interface to manage the process.
- Improving consistency and quality in execution of test.
- Re-configuring the portal as per the availability of test environments, products and databases.

## 1.7 Thesis Organization

- **Chapter 1:** This chapter introduces basics of both manual and automated testing. It also introduces the basic methodology about how one can test the quality of software. It also describes different types of testing tools available for automation testing.
- **Chapter 2:** Describes the fundamentals required for this thesis. It describes the Significance of Validation Phase in the Integration Testing. It also describes the review of the existing Retail Applications so that a better Integration Testing can be achieved.
- **Chapter 3:** This chapter defines the Integration problem and the way by we are dealing it in the thesis. It also describes the main workflow.
- **Chapter 4:** This chapter explains the different tools and utilities that we have used for a better understanding of results and to achieve a better regression testing.
- **Chapter 5:** In this chapter the results of thesis are discussed and also the main reasons behind the not so good quality of applications are listed.
- **Chapter 6:** This chapter states the conclusion and the scope for future work.

# Chapter 2

# Literature Review

## 2.1 Automated Testing

Automation testing tools are used for automated testing. The main task of automation testing tool is to test the intended condition and reduce human efforts. It is much faster than manual testing and also more reliable as each test case is performed with precision. Software quality and reliability are increased with automation testing [7]. In automation testing we can write sophisticated test cases to detect hidden defects which is not possible in manual testing.

### 2.1.1 Testing Life Cycle

Testing life cycle may vary from organization to organization but, there is a typical cycle for testing [8]. The below cycle is used in organizations which uses the Waterfall development model. The same practices are also found in other development models, but might not be as clear or explicit.

- **Requirements analysis:** In the requirements phase itself, the testing of the software should began. In the design phase, two things can be determined namely, what aspects of a design are testable and with what parameters those tests work.
- **Test planning:** Test strategy, test plan, testbed creation. A plan is needed as number of activities will be carried out during testing.
- **Test development:** Test scenarios, test procedures, test scripts, test datasets, test cases to use in testing software.
- **Test execution:** Proper planning and test documentation is done and then testers execute the software and if there is any error they report to development team.
- **Test reporting:** Testers makes metrics and also the final reports after the testing is completed on their test effort and also ensure if the software is ready to be released or not.

- **Test result analysis:** Test result analysis or defect analysis, is done by the development team usually along with the client, so that we can ensure what defects needs to be fixed, assigned, deferred (left for dealing in future) or rejected (software is fit to use).
- **Defect Retesting:** Once defects are fixed by development team, the software is again tested for the proper functioning.
- **Regression testing:** It is common to have a small test program built of a subset of tests, for each integration of new, modified, or fixed software, in order to ensure that the latest delivery has not ruined anything and that the software product as a whole is still working correctly.
- **Test Closure:** Once the test meets the exit criteria, the activities such as capturing the key outputs, lessons learned, results, logs, documents related to the project are archived and used as a reference for future projects.

### 2.1.2 Different Types of Automated Testing

- **Automated Web Testing:** Software testing that uses automation for focusing on web applications is called automated web testing. Number of issues can be addressed before the system is revealed to the public by complete testing of a web-based system before going live [21]. Issues can be operating systems, devices, and its ability to adapt to the multitude of desktops, its accessibility to handicapped users and fully able users, the basic functionality of the site and the security of the web application.
- **Automated GUI Testing:** Graphical user interface testing is the way of testing a product's graphical user interface to ensure it meets its specifications. Variety of test cases are used in this.
- **Automated Unit Testing:** Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors [9]. Depending on the organization's expectations for software development, unit testing might include static code analysis, data-flow

analysis, metrics analysis, peer code reviews, code coverage analysis and other software testing practices.

- **Automated Integration Testing:** Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system [10].

There are a number of open source testing tools [20, 23] available in the software market. Although the core functions of these tools are similar (depending on the automation case), they differ in functionality, features, usability [11]. The parameters that one looks for comparison of tools are,

- Recording Efficiency
- Capability of generation of scripts
- Data driven testing
- Test result reports
- Reusability
- Execution speed
- Playback of the scripts
- Easy to Learn
- Cost

## 2.2 Introduction to EIT

Enterprise Integration Tools and Utilities are used by EIT members to ease their integration work. The major functionalities that comes under this are, EIT Tools, EIT Task Management, EIT Dashboard and EIT Reports with multiple utilities under each of them. Graphical representation of analysis is shown in EIT Dashboard. Reporting tools for messages covers, product traceability matrix and foundation matrix are available under EIT Reports. Functions like Configuration Management, Data Verification and Transaction Validation lies under EIT Tools section.

## 2.3 Enterprise Products

If we don't know what is really happening behind the scene then it's totally of no use of automating the things as we don't know what should happen after what. So understanding the characteristics of automation as well as retail cycle together is very important. This will help in better analysis of result.

A brief overview of retail applications, test scenarios, framework and technologies are necessary for a good quality of automation. Automated test cases should be reusable and maintainable. Some of the retail applications that were used in this project are as follows,

- **RMS –** RMS stands for Retail Merchandising System [12]. It acts as the central repository for all the other applications in retail world i.e. if any kind of update is made in any application then it's obvious that change will be reflected in RMS. It act as an admin application that have access to functionality of all other retail applications available. It also makes sure that data is integrated by using the Retail Integration Bus. It provides a smooth environment so that different applications can communicate and helps in taking critical decisions. Oracle have dedicated applications for different retail tasks like Sales, Analysis, Warehousing and many more. RMS acts as a heart of all the above mentioned products.
- **SIM –** SIM stands for Store Inventory Management [13]. This application is basically used for keeping a track of stock of good in different stores. The stock changes after every task is performed like, Ordering, Returning and Transfers. The stock for the good changes accordingly and SIM keeps a track for that. In short it tracks shipping and receiving of goods.
- **WMS –** WMS stands for Warehouse Management System [14]. All the operations related to warehousing are taken care by this application. Some of the operations that are done in WMS are picking, shipping, receiving and block storage of goods. Every location (store) has a unique warehouse for it that stores the goods that cannot be kept in stores itself.
- **ReIM –** ReIM stands for Retail Invoice Matching [15]. Each Order and RTV (Return to Vendor) scenarios have invoice which consists for cost and quantity. The main task of ReIM is to verify that cost and quantity from the invoice. If we

find any mismatching in the cost and the quantity then it's sent for review and both cost and quantity are matched.

- **RPM –** RPM stands for Retail Price Management [16]. This application is used for setting the price for a particular item. It is also used to declare a sale or extra discount on the item. Promotions and Clearance are created on item by zone i.e. sale is declared in a particular area. The sale can be amount off or percent off.
- **Allocation –** Allocation [17] can be used for distributing items from different types of sources to a single location. It's a kind of transfer which includes shipping from multiple stores and receiving at single store. It can be initiated by transfer, warehouse, PO etc. Scheduled Allocations, What if Allocation and Standard Allocation are some kind of allocations.

## 2.4 RIB

A require a medium through which different retail applications can communicate and share data with each other. This feature is facilitated by Retail Integration Bus (RIB) [18]. RIB is there for all products like RMS, SIM, WMS etc. as each of them have to share data according to the scenario as in Figure 2.1. There are two adapters in this, namely, Publishers and Subscribers. Publishers are used for publishing the data onto the RIBs which will then be accessible to the application for which it was published. For publishing the data onto RIBs the Publish adapter must be up and working. Subscriber is used for subscribing i.e. taking the data from the RIBs. If one application has to take some data from the RIBs then its Subscriber adapter must be up and working.

The functionalities that are provided by RIBs include support for fast delivery of retail volumes, ability to fix and retry message, real time messaging, guaranteed sequential delivery within a message family regardless of errors, guaranteed once only delivery and multi-threading. Due to all of these an infrastructure that is highly available and highly reliable is created.

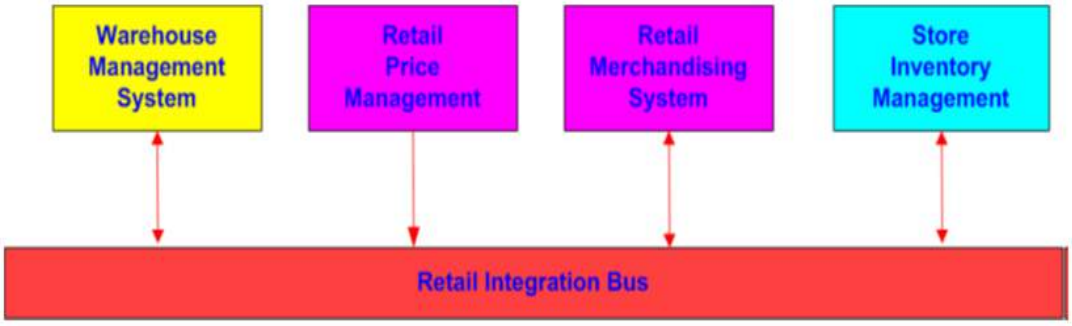



Figure 2.1 Retail Integration Bus

## 2.5 Foundation Data

Foundation Data is the pre required data that is needed before performing any scenario. This data can include information about Items, Warehouses, Suppliers and Stores. We need this data because for performing any scenario we must have some Stores, Warehouses, Suppliers and Items in hand or else we won't be able to perform any scenario. This data also include the information about the stocks at different places as well as different types of Stores and Items.

There are many changes as per the requirement in applications. So applications must be deployed regularly. Database and basic transactions should be tested after each releases. QA environments provides details about applications, integrations and databases.

| Supplier | Warehouse | Store | Item |
|---|---|---|---|
| • Supplier can be manufacturer of item<br>• Located in various countries | • Retailers like Big Bazar have their warehouse to keep stock of items in bulk<br>• Items are transferred to various stores | • Located in countries<br>• Various types of items | • Various kind of items are stored in stores<br>• Items are identified uniquely in stores |

Figure 2.2 Foundation Data

# Chapter 3

# Architecture and Workflow

## 3.1 Workflow to Achieve Objectives

The overall flow of the process start from executing test cases to different applications involved and the results can be seen from Figure 3.1.

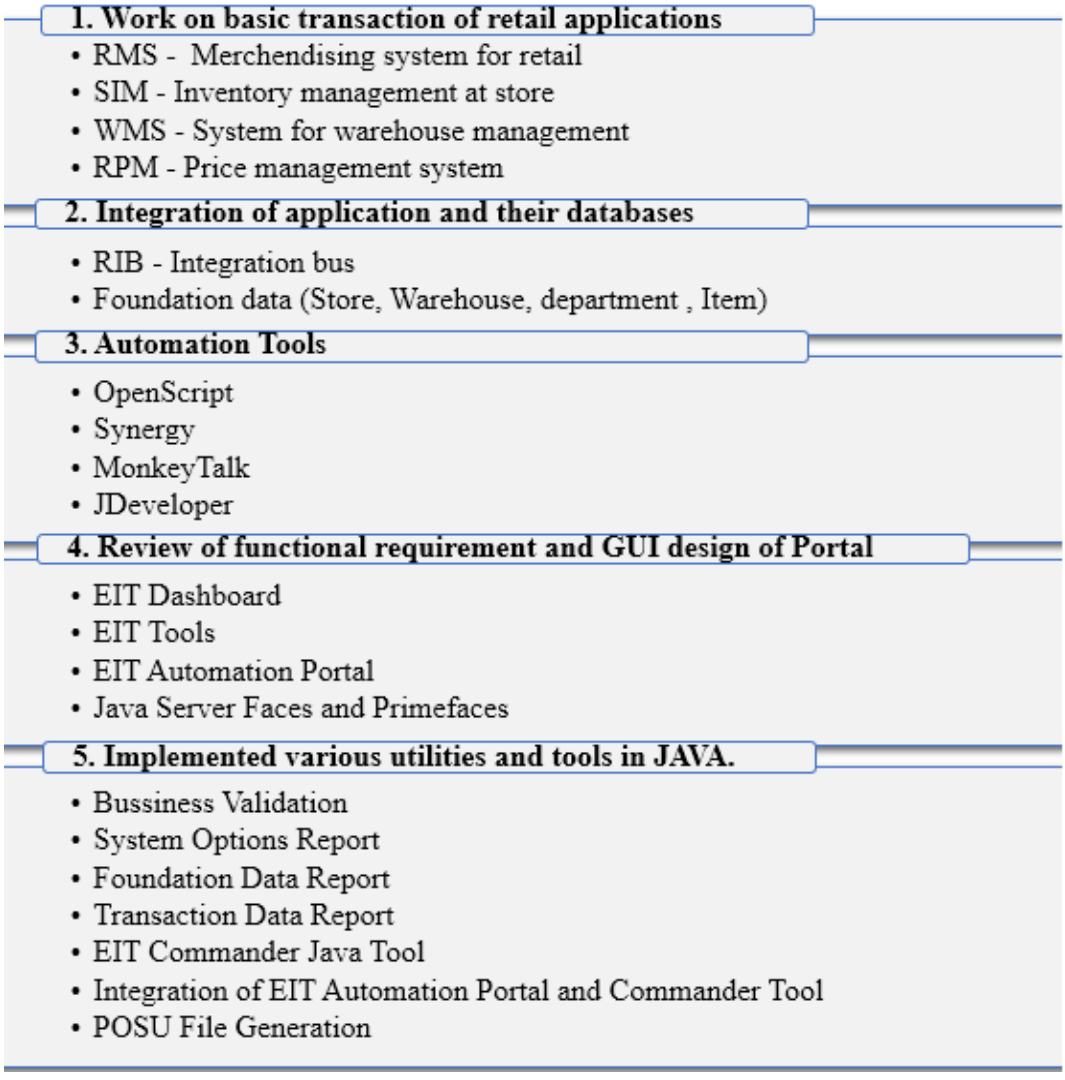



Figure 3.1 Overall Workflow

At very initial stage, a particular test case is executed manually for a better understanding of the scenario. Then while automating the case, the integration services (RIBs) are taken

into consideration. Sometimes due to one scenario the RIBs may get down thus proper precautions are taken. The tools that were used for automating the scripts depend on the nature of the scenario i.e. the different applications used in. For example, A Warehouse to Warehouse Transfer only need the RIBs (both Subscriber and Publisher) for both Warehouse and RMS to be up and working. Subscriber usually subscribes the data information from the RIB whereas Publishers publishes the data information into the RIB. In this case only OpenScript is used as Warehouse application is Form based application. If SIM is in the picture, suppose a Store to Store Transfer then one have to use OpenScript as well as MonkeyTalk and JDeveloper for automating this scenarios as SIM application is built in Mobile Application Framework. After the script is ran completely one can check the results from the EIT Dashboard which shows the exact reason for the script failure. After the cause of failure of script is known, the same is checked with Dev team and required steps are taken so that the software quality can be improved. The utilities that are created are used for the better and easy flow of the test execution. These utilities gives us the pre required data that is needed for the execution of the scenario.

## 3.2 Automation Scenario

OpenScript was used for the ADF application and forms, Synergy is used for java applications and MonkeyTalk for Mobile Application Framework applications. After each stage of the scenarios, database validations and product UI validations are performed to check the transaction is performed successfully or not.

Tool used for the automation of specific product:
RMS, WMS, Allocation, RPM – **OpenScript**
SIM – **Synergy**
SIM-MAF – **MonkeyTalk and JDeveloper**
For this thesis the following scenarios [12] has been executed,

- Purchase Order (PO)
- Direct Store Delivery (DSD)
- X-Order
- Item Request

- Transfer
- X-Transfer
- RUA
- Replenishment
- Return to Vendor (RTV)
- Commerce Anywhere (CA)
- Cross Channel (XC)
- RCA
- Mass Return Transfer (MRT)

Some of the scenarios are described below.

### 3.2.1 Purchase Order (PO)

PO is a scenario using which the order for the item by particular warehouse will be placed and inventory will be provided to that warehouse by the supplier. In this scenario, PO will be created in RMS and then in WMS appointment will be created with type PO and appointment will be received. There are different types based on the appointment received after edit, appointment deleted, PO cancelled. Products included in this scenario: RMS, SIM and WMS.

### 3.2.2 Direct Store Delivery (DSD)

DSD is a kind of purchase order in which warehouse doesn't come into picture. Once the order is created in RMS, the same is received directly in the destination store i.e. the quantity is shipped directly from supplier to the store.

### 3.2.3 Item Request

In Item Request scenario, item request is created in SIM which will basically trigger a PO for the item. So after creating item Request, the PO which is generated for that item should be received in SIM and validate in RMS and SIM UI and DB. There are different scenarios

of Item Request based on the item is having source as warehouse or supplier, Item Request with UOM case or unit. Products included in this scenario: RMS and SIM.

### 3.2.4 Transfer

In transfer scenario, the item will be transferred from one location to the other so quantity of item will be decreased at one location and will be increased at other. There are total four types of transfers:

- **Store to Store** - The transfer can be RMS initiated or SIM initiated that is it can be created in RMS or SIM. Source and Destination of transfer will be store. Transfer will be shipped from source store and it will be received in destination store in SIM. Products included in this scenario: RMS and SIM.
- **Store to Warehouse** - The transfer can be RMS initiated or SIM initiated that is it can be created in RMS or SIM. Source of transfer will be store whereas destination of transfer will be warehouse. Transfer will be shipped from source store in SIM and it will be received in destination warehouse in WMS by using appointment type ASN. Products included in this scenario: RMS, SIM and WMS.
- **Warehouse to Store** - The transfer can be RMS initiated or WMS initiated that is it can be created in RMS or WMS. Source of transfer will be warehouse whereas destination of transfer will be store. Transfer will be shipped from source warehouse in WMS using trailer and it will be received in destination store in SIM. Products included in this scenario: RMS, SIM and WMS.
- **Warehouse to Warehouse** - The transfer can be RMS initiated or WMS initiated that is it can be created in RMS or WMS. Source of transfer will be warehouse whereas destination of transfer will be warehouse. Transfer will be shipped from source warehouse in WMS using trailer and it will be received in destination warehouse in WMS by using appointment type ASN. Products included in this scenario: RMS and WMS.

### 3.2.5 Return to Vendor

Return to Vendor (RTV) is the scenario in which the stocks are returned back to the supplier from which it was received earlier. This return can be from a warehouse or store. One can return any amount of quantity he want to return or even the entire stock for an particular item.

### 3.2.6 Replenishment

Replenishment is an order for additional goods from a warehouse or supplier, in order to replenish depleted stock in a store or warehouse. Replenishment in RMS allows retailers to set up the automatic ordering of items, and RMS can monitor the inventory positions at locations throughout a retail enterprise, down to the item/location levels. Depending on the algorithm used, RMS replenishment can be configured to make recommendations, which can be manually added to a purchase order or transfer, or it can create purchase orders or transfers directly, depending on the level of automation desired.

## 3.3 Detailed Example Depicting the Flow and Validation Procedure

### 3.3.1 Scenario

**Store (A) to Store (B) Transfer initiated through Retail Merchandising System**

In this case the things that we are worried about the initial and final stocks at both the stores (A & B). The final stock for Stock A should be decreased by the quantity requested whereas the stock values for Store B should increase by the requested quantity. As this scenario includes Store hence OpenScript as well as JDeveloper are the software that will be needing for its automation.

The detailed step by step execution of Store to Store transfer is shown in Figure 3.2, it gives an overview about the tables that have to be checked after every step.

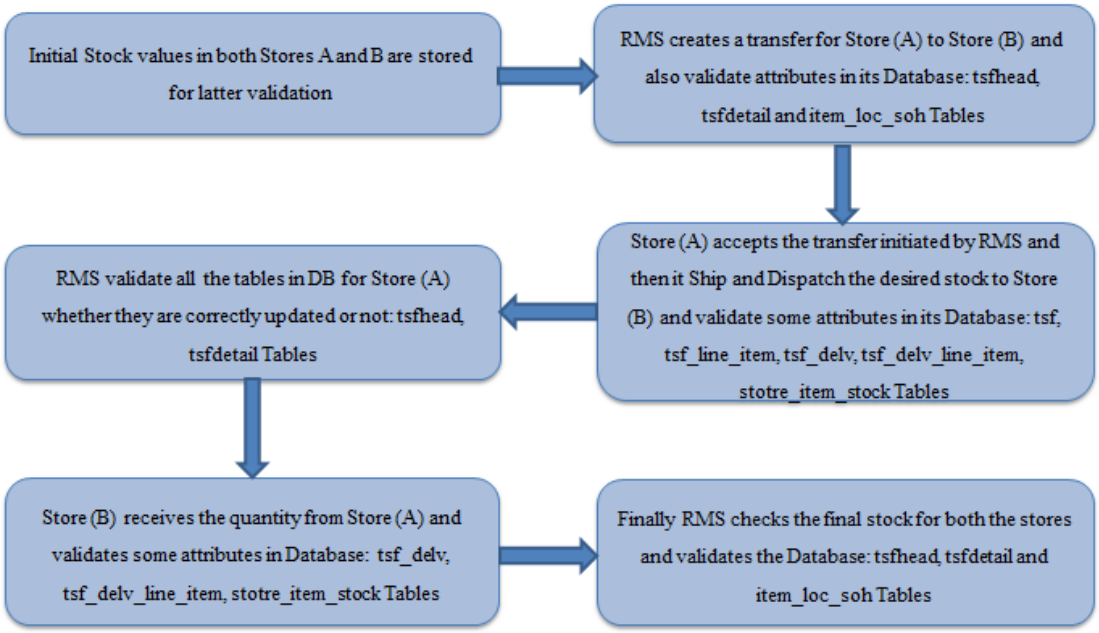



Figure 3.2 Store (A) to Store (B) Transfer initiated through RMS

As in this scenario OpenScript and JDeveloper are used for automation process, below is the detailed description of what is done at every step,

All the scripts that have been written for automation testing have a driver code which is mostly written in Openscript. The driver code has number of functions defined which are called at the required time. How different functions are called (in this scenario, three different functions) when required is shown in Figure 3.3. The order in which functions are called is,

- **createTransfer() -** This function is responsible for creating Transfer for Store A to Store B of the desired quantity.
- **validatePostShipment() -** This function validates the value after transfer is shipped from Store A to Store B.
- **validatePostReceipt() -** This function validates the value after transfer is received at Store B.

```
public void run() throws Exception {

    try{


        Map<String,String> inputParamsMap = EIT_Util.prepareTestDataMap(getSettings().get("customParameters"));

        String testCaseFilePath = getScriptPackage().getRepositoryPath();
        //String propFilePath = testCaseFilePath.substring(0,testCaseFilePath.lastIndexOf("\\"))+"\\Environments\\RMS.properties";
        String propFilePath = testCaseFilePath+"\\Environments\\RMS.properties";
        Properties prop = EIT_Util.getProductProperties(propFilePath);

        if(null!=prop){
            inputParamsMap.put("rmsURL",prop.getProperty("RMS_URL"));
            inputParamsMap.put("rmsUser",prop.getProperty("RMS_Login_User"));
            inputParamsMap.put("rmsPassword",prop.getProperty("RMS_Login_Pwd"));
        }

        System.out.println(" ##### inputParamsMap from open script 728962_27252_RMS.SIM.New.All Item Types.Store to Store Intercompany Transfer ##### ..."+inputParamsMap);

        if(null== inputParamsMap.get("initialSohFromLoc") || "".equals(inputParamsMap.get("initialSohFromLoc")))
        {
            createTransfer(inputParamsMap,prop);

        }else if(null== inputParamsMap.get("received") || "".equals(inputParamsMap.get("received")))
        {
            validatePostShipment(inputParamsMap,prop);
        }
        else{
            validatePostReceipt(inputParamsMap,prop);
        }

    }
    catch(Exception e){
        System.out.println("Exception in open script Create PO for Regular Item....."+e);
    }
```

Figure 3.3 OpenScript code snapshot

The view of code from JDeveloper side which act as the driver code for the MonkeyTalk part for Dispatching is shown in Figure 3.4. It calls the required script at the desired time. Here it calls different main scripts in the order,

- **accept.mt** – This script is responsible for accepting the transfer from Store A to Store B at Store A. This is important because its RMS initiated. The initiation store first checks if it's having the stocks that will satisfy the ordered quantity or not. If condition is satisfied then only further execution is done else script fails.
- **ship.mt** – This script is responsible for shipping the container with the ordered quantity.
- **Dispatch.mt** – This script is used for dispatching the container to Store B location.
- **beforeTsfValidations.js, postAcceptValidation.js and ConditionalLogin.js** are used for validations in between process and for login into the application.

RMS_initiated_All_Item_Types_Store_to_Store_Manual_Requisition_Transfer_Ship_Full_and_Receive_1.mt

| Row | Component | Monkey Id | Action | Arguments |
|---|---|---|---|---|
| 1 | Vars | * | Define | count itemId1 itemId2 transferQty adjustQty locType fromL... |
| 2 | Script | beforeTsfValidation.js | Run | 0 ${itemId1} ${itemId2} ${transferQty} ${locType} ${from... |
| 3 | App | * | Restart | |
| 4 | Script | postAcceptValidation.js | Run | ${transferId} 0 ${itemId1} ${itemId2} ${transferQty} ${loc... |
| 5 | #Script login.mt Ru... | | | |
| 6 | Script | ConditionalLogin.js | Run | |
| 7 | Script | menu_SIM.mt | Run | |
| 8 | Script | selectstore.mt | Run | ${fromLoc} |
| 9 | Script | menu_SIM.mt | Run | |
| 10 | Script | accept.mt | Run | |
| 11 | Script | menu_SIM.mt | Run | |
| 12 | Script | beforeTsfValidation.js | Run | 1 ${itemId1} ${itemId2} ${transferQty} ${locType} ${from... |
| 13 | Script | ship.mt | RunWith | SIMtsfID.csv |
| 14 | Script | beforeTsfValidation.js | Run | 2 ${itemId1} ${itemId2} ${transferQty} ${locType} ${from... |
| 15 | Script | Dispatch.mt | RunWith | shipmentID.csv |
| 16 | Script | menu_SIM.mt | Run | |
| 17 | Script | logout.mt | Run | |
| 18 | Script | postAcceptValidation.js | RunWith | data1.csv |

Figure 3.4 JDeveloper code snapshot for  Dispatching Quantity

The view of code from JDeveloper side which act as the driver code for the MonkeyTalk part for Receiving is shown in Figure 3.5. It calls the required script at the desired time. Here it calls different main scripts in the order,

- **receive.mt** – This script is responsible for receiving the ordered quantity at destination store B.
- **receivingValidation.js and ConditionalLogin.js** are used for validations after receiving process and for login into the application.


RMS_initiated_All_Item_Types_Store_to_Store_Manual_Requisition_Transfer_Ship_Full_and_Receive_2.mt

| Row | Component | Monkey Id | Action | Arguments |
|---|---|---|---|---|
| 1 | App | * | Restart | |
| 2 | Vars | * | Define | transferId count itemId1 transferQty adjQty locType fromL... |
| 3 | #Script login.mt Ru... | | | |
| 4 | Script | ConditionalLogin.js | Run | |
| 5 | Script | menu_SIM.mt | Run | |
| 6 | Script | selectstore.mt | Run | ${toLoc} |
| 7 | Script | menu_SIM.mt | Run | |
| 8 | Script | receive.mt | RunWith | data.csv |
| 9 | Script | menu_SIM.mt | Run | |
| 10 | Script | logout.mt | Run | |
| 11 | Script | receivingValidation.js | RunWith | data2.csv |

Figure 3.5 JDeveloper code snapshot for Receiving Quantity

### 3.3.2 Validations at Every Step

The attributes [12,13] that are being validated in different tables are as follow:

**Tsfhead table**

In this table the below listed attributes are validated after every step with the following values,

Table 3.1 Validation of attributes in tsfhead (RMS)

| Stages | Tsf_no | From_loc | To_loc | Status |
|---|---|---|---|---|
| Approved | Constant | Constant | Constant | **A** |
| Shipped | Constant | Constant | Constant | **S** |
| Dispatched | Constant | Constant | Constant | **D** |
| Received | Constant | Constant | Constant | **R** |

**Tsfdetail table**

In this table the below listed attributes are validated after every step with the following values,

Table 3.2 Validation of attributes in tsfdetail (RMS)

| **Stages** | **Tsf_no** | **Item** | **Tsf_qty** | **Ship_qty** | **Received_qty** |
|---|---|---|---|---|---|
| Approved | Constant | Constant | Constant | **0** | **0** |
| Shipped | Constant | Constant | Constant | **Quantity** | **0** |
| Dispatched | Constant | Constant | Constant | **Quantity** | **0** |
| Received | Constant | Constant | Constant | **0** | **Quantity** |

**Item_loc_soh tables**

In this table the below listed attributes are validated after every step with the following values,

On Store (A) side,

Table 3.3 Validation of attributes in item_loc_soh for Store A (RMS)

| **Stages** | **Item** | **Loc** | **Stock_on_ha-nd** | **In_transi-t_qty** | **Tsf_reserved_-qty** |
|---|---|---|---|---|---|
| Approved | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Shipped | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Dispatch-ed | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Received | Constant | Constant | **Previous - Quantity** | Previous | **Previous** |

On Store (B) side,

Table 3.4 Validation of attributes in item_loc_soh for Store B (RMS)

| **Stages** | **Item** | **Loc** | **Stock_on_h-and** | **In_transit_-qty** | **Tsf_reserve-d_qty** |
|---|---|---|---|---|---|
| Approved | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Shipped | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Dispatch-ed | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Received | Constant | Constant | **Previous + Quantity** | **Previous** | Previous |

**Tsf table**

In this table the below listed attributes are validated after every step with the following values,

Table 3.5 Validation of attributes in tsf (SIM)

| **Stages** | **External_id** | **Destination_id** | **Status** |
|---|---|---|---|
| Approved | Constant | Constant | **1** |
| Shipped | Constant | Constant | **2** |
| Dispatched | Constant | Constant | **7** |
| Received | Constant | Constant | **9** |

**Tsf_line_item Table**

In this table the below listed attributes are validated after every step with the following values,

Table 3.6 Validation of attributes in tsf_line_item (SIM)

| Stages | Tsf_no | Item | Quantity_requested | Quantity_received |
|---|---|---|---|---|
| Approved | Constant | Constant | **Quantity** | **0** |
| Shipped | Constant | Constant | **Quantity** | **Quantity** |
| Dispatched | Constant | Constant | **Quantity** | **Quantity** |
| Received | Constant | Constant | **Quantity** | **Quantity** |

**Store_item_stock tables**

In this table the below listed attributes are validated after every step with the following values,

On Store (A) side,

Table 3.7 Validation of attributes in item_loc_soh for Store A (SIM)

| Stages | Item_id | Store_id | Stock | In_transit_qty | Tsf_reserved_qty |
|---|---|---|---|---|---|
| Approved | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Shipped | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Dispatched | Constant | Constant | **Previous** | Previous | **Previous + Quantity** |
| Received | Constant | Constant | **Previous - Quantity** | Previous | **Previous** |

On Store (B) side,

Table 3.8 Validation of attributes in item_loc_soh for Store B (SIM)

| **Stages** | **Item_id** | **Store_id** | **Stock** | **In_transit_qty** | **Tsf_reserved_qty** |
|---|---|---|---|---|---|
| Approved | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Shipped | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Dispatched | Constant | Constant | **Previous** | **Previous + Quantity** | Previous |
| Received | Constant | Constant | **Previous + Quantity** | **Previous** | Previous |

If everything works as desired then these should be the validation values. As values are being validated after every step, thus if anything goes wrong one came to know immediately. The reason behind the un-wanted behavior are mainly be due to RIBs issues, application server side issues, Web Service issues. These issues can be easily seen through EIT Dashboard.

# Chapter 4

# Features and Implementation

## 4.1 Automation Portal and Controller

- One Click Automation – Instead of launching one-one script from a tool (OpenScript, MonkeyTalk or Synergy), launching it in a bunch easily by just selecting the scripts from a Portal.
- Constant Execution against live environment – Can be used for overnight execution against specific environment.
- Publish Execution Results email – Result of execution should be sent to the team as and when the execution completes by email.

### 4.1.1 Requirement for Portal

- To run OpenScript & Synergy on same machine – Previously the automation team was using two machines to run a script because the tools use different java for functioning. So using 2 machines was wastage of resource.
- On demand run for cloud and On Prem Env's – For cloud environment, DB access will not be there so need to add UI validations.
- On demand run for selected test case – Easy selection and launching of test script.
- Track the current execution – Can track the live execution of test scripts.
- Historic view of test case run results – Can access the old results and view it in a report mode as well chart mode.

### 4.1.2 Portal and Controller Design

The main structure of Portal can be seen from Figure 4.1. Its main components are Master Spreadsheet, Integration Hub, Property Files and Controller Program. Few characteristics of Portal can are listed below.

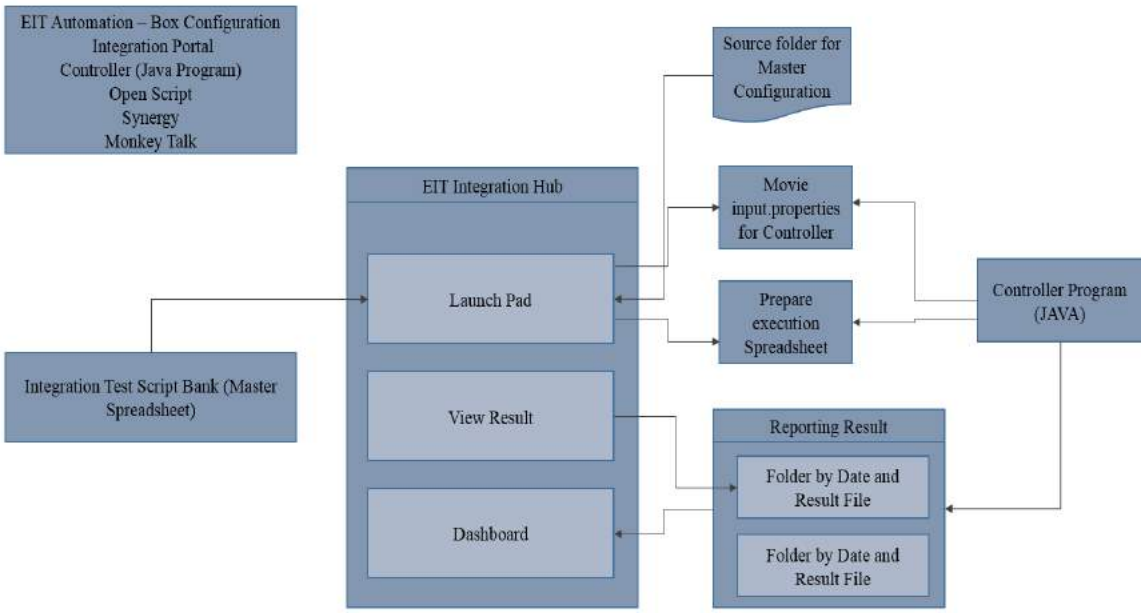
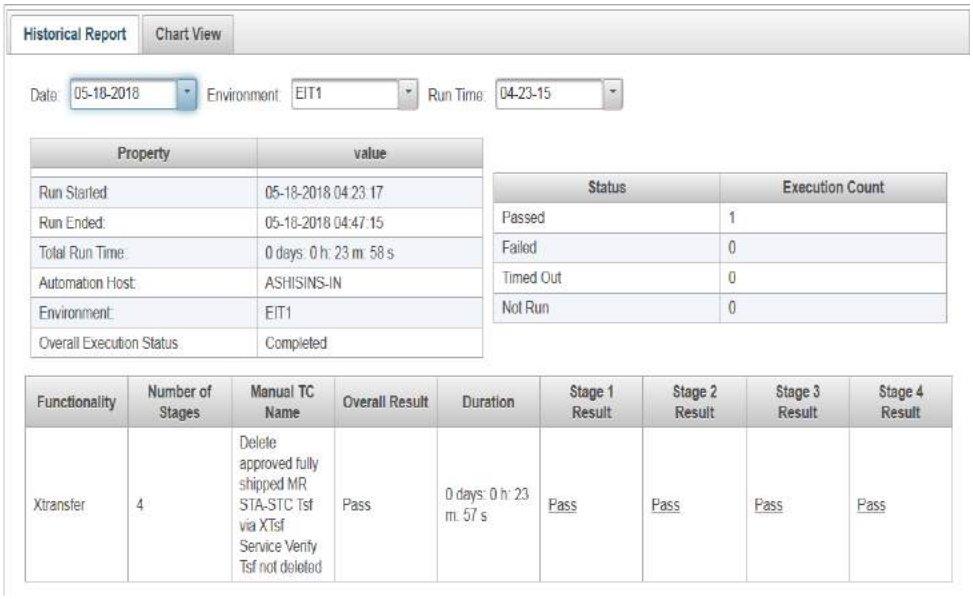



Figure 4.1 Portal and controller integration

- Read the file having test case name, data, path, etc. Read two excel file named "Execution_Commander.xlsx" and "Execution_Report.xlsx". Execution_Commander file is used to read the test case path and data whereas Execution_Report file will be having the result of execution. (Stage wise result, Overall result, total Runtime, total TC passed & failed as shown in Figure 4.2)
- Read the environment file against which the test case will be executed. Read the properties file move the product specific property file to OpenScript and Synergy Environments folder which will be used while execution and the same can be seen from Figure 4.3.
- Run the OpenScript and Synergy on same machine. Synergy uses the JAVA_TOOL_OPTIONS env. Variable which cannot be used for OpenScript so before starting the OpenScript, controller will remove this env. Variable.
- Parse the Result xml file of OpenScript and synergy to get the result. To get the final result, controller will parse the tags of result xml file and put the result (Pass/Fail) in Execution_Report. Save the result of stages in file.
- Publish Email Result. After the completion of execution, the result will be sent to the team.



Historical Report | Chart View

Date: 05-18-2018 Environment: EIT1 Run Time: 04-23-15

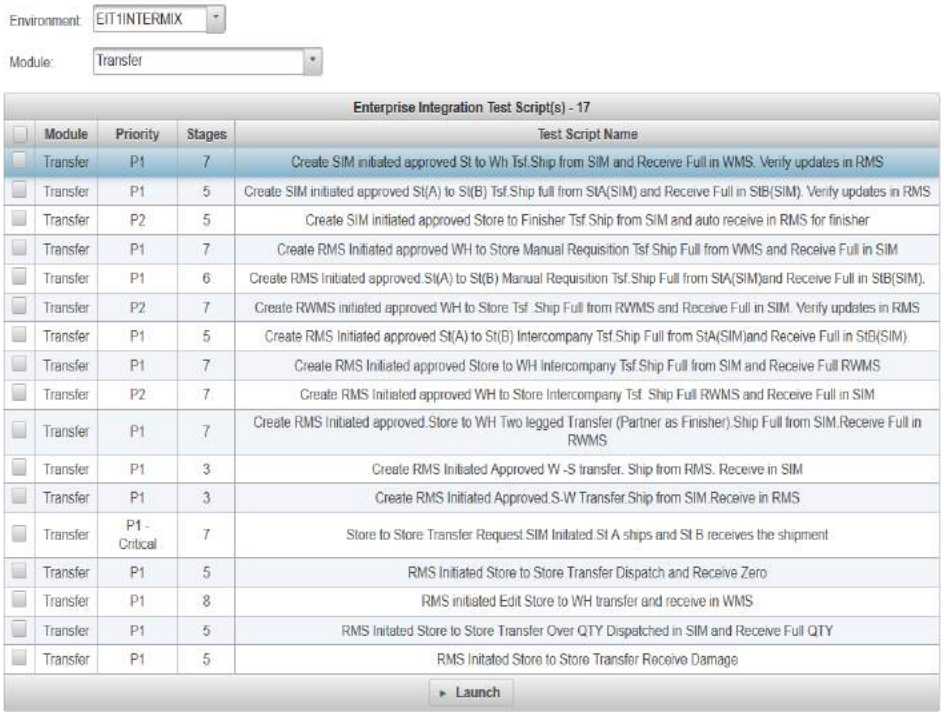
| Property | value |
|---|---|
| Run Started: | 05-18-2018 04:23:17 |
| Run Ended: | 05-18-2018 04:47:15 |
| Total Run Time: | 0 days: 0 h: 23 m: 58 s |
| Automation Host: | ASHISINS-IN |
| Environment: | EIT1 |
| Overall Execution Status: | Completed |

| Status | Execution Count |
|---|---|
| Passed | 1 |
| Failed | 0 |
| Timed Out | 0 |
| Not Run | 0 |

| Functionality | Number of Stages | Manual TC Name | Overall Result | Duration | Stage 1 Result | Stage 2 Result | Stage 3 Result | Stage 4 Result |
|---|---|---|---|---|---|---|---|---|
| Xtransfer | 4 | Delete approved fully shipped MR STA-STC Tsf via XTsf Service Verify Tsf not deleted | Pass | 0 days: 0 h: 23 m: 57 s | Pass | Pass | Pass | Pass |

Figure 4.2 Detailed description of result after execution

- In case of failure of initial stages, block the rest of stages and fail that test case.



Environment: EIT1INTERMIX

Module: Transfer

Enterprise Integration Test Script(s) - 17

| Module | Priority | Stages | Test Script Name |
|---|---|---|---|
| Transfer | P1 | 7 | Create SIM initiated approved St to Wh Tsf.Ship from SIM and Receive Full in WMS. Verify updates in RMS |
| Transfer | P1 | 5 | Create SIM initiated approved St(A) to St(B) Tsf.Ship full from StA(SIM) and Receive Full in StB(SIM). Verify updates in RMS |
| Transfer | P2 | 5 | Create SIM initiated approved Store to Finisher Tsf.Ship from SIM and auto receive in RMS for finisher |
| Transfer | P1 | 7 | Create RMS Initiated approved WH to Store Manual Requisition Tsf.Ship Full from WMS and Receive Full in SIM |
| Transfer | P1 | 6 | Create RMS Initiated approved.St(A) to St(B) Manual Requisition Tsf.Ship Full from StA(SIM)and Receive Full in StB(SIM). |
| Transfer | P2 | 7 | Create RWMS initiated approved WH to Store Tsf .Ship Full from RWMS and Receive Full in SIM. Verify updates in RMS |
| Transfer | P1 | 5 | Create RMS Initiated approved St(A) to St(B) Intercompany Tsf.Ship Full from StA(SIM)and Receive Full in StB(SIM) |
| Transfer | P1 | 7 | Create RMS Initiated approved Store to WH Intercompany Tsf.Ship Full from SIM and Receive Full RWMS |
| Transfer | P2 | 7 | Create RMS Initiated approved WH to Store Intercompany Tsf .Ship Full RWMS and Receive Full in SIM |
| Transfer | P1 | 7 | Create RMS Initiated approved.Store to WH Two legged Transfer (Partner as Finisher).Ship Full from SIM.Receive Full in RWMS |
| Transfer | P1 | 3 | Create RMS Initiated Approved W -S transfer. Ship from RMS. Receive in SIM |
| Transfer | P1 | 3 | Create RMS Initiated Approved.S-W Transfer.Ship from SIM.Receive in RMS |
| Transfer | P1 - Critical | 7 | Store to Store Transfer Request.SIM Initated.St A ships and St B receives the shipment |
| Transfer | P1 | 5 | RMS Initiated Store to Store Transfer Dispatch and Receive Zero |
| Transfer | P1 | 8 | RMS initiated Edit Store to WH transfer and receive in WMS |
| Transfer | P1 | 5 | RMS Initated Store to Store Transfer Over QTY Dispatched in SIM and Receive Full QTY |
| Transfer | P1 | 5 | RMS Initated Store to Store Transfer Receive Damage |

Launch

Figure 4.3 Requirements for executing a test case through automation

## 4.2 Application Status Check

Application status check task is to facilitate in the automation of nightly scheduled run on daily basis only if the applications that are in use have their RIBs up and working properly. For now we were kicking off the nightly run in different production servers without checking the status of RIBs, due to which the failure counts were increasing. To compensate this, application status check utility is developed.

### 4.2.1 Table Requirements

Table nam "Automation_config_check" was designed with the following fields,

- **Environment –** It shows the running environment of test scripts like EIT1, EIT2, QA7, QA8 etc.
- **HOST –** It shows the name of production server from where the run in kick off.
- **RUN_DATE –** This field stores the date of run of configuration scripts.
- **RMS, RWMS, SIM, RPM, ALLOCATION, RI, RDE, RESA, RMS_WEB, IGS_WEB, RIB, REIM –** These are fields for different application which stores UP/DOWN according to the status of their RIBs.

The snapshot of Automation_config_check table with different fields is show in Figure 4.4.

| | ENVIRONMENT | HOST | MODULE | RMS | RWMS | SIM | RPM | RUN_DATE | ALLOCATION | RI | RDE | RESA | RMS_WEB | IGS_WEB | SIM_WEB | RIB | REIM | NIGHTLY_BATCH |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 86 | EIT2 | Blr2223131 | (null) | DOWN | (null) | (null) | DOWN | 11-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 85 | EIT1 | EIT_Auto_Prod-1 | (null) | UP | UP | UP | UP | 11-NOV-17 | UP | UP | UP | UP | UP | UP | (null) | UP | UP | UP |
| 84 | EIT2 | Blr2223131 | (null) | DOWN | (null) | (null) | DOWN | 10-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 83 | EIT1 | EIT_Auto_Prod-1 | (null) | UP | UP | UP | UP | 10-NOV-17 | DOWN | UP | UP | UP | UP | UP | (null) | UP | UP | UP |
| 82 | EIT2 | Blr2223131 | (null) | DOWN | (null) | (null) | DOWN | 09-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 81 | EIT1 | Blr2223131 | (null) | UP | UP | UP | UP | 09-NOV-17 | DOWN | UP | UP | UP | UP | UP | (null) | UP | UP | UP |
| 80 | EIT2 | Blr2223131 | (null) | DOWN | (null) | (null) | DOWN | 08-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 79 | EIT1 | EIT_Auto_Prod-1 | (null) | UP | UP | UP | UP | 08-NOV-17 | DOWN | UP | UP | UP | UP | UP | (null) | UP | UP | (null) |
| 78 | EIT2 | Blr2223131 | null | DOWN | (null) | (null) | DOWN | 07-NOV-17 | DOWN | UP | UP | DOWN | UP | UP | UP | UP | DOWN | DOWN |
| 77 | EIT1 | Blr2223131 | null | UP | UP | UP | UP | 07-NOV-17 | UP | UP | UP | UP | UP | UP | (null) | UP | UP | UP |

Figure 4.4 Data in Automation_config_check table

### 4.2.2 Constraints for Table

“Automation_config_check” table gets updated only if RIB.properties is having “Integration_flg” as “Yes”. How to set integration_flag variable value can be seen in Figure 4.5. Only one entry is made for particular environment in a day and multiple run would simply update their status.

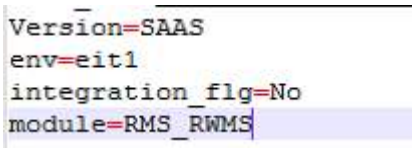



Figure 4.5 Some fields from RIB.properties file

### 4.2.3 Requirements of Bat File

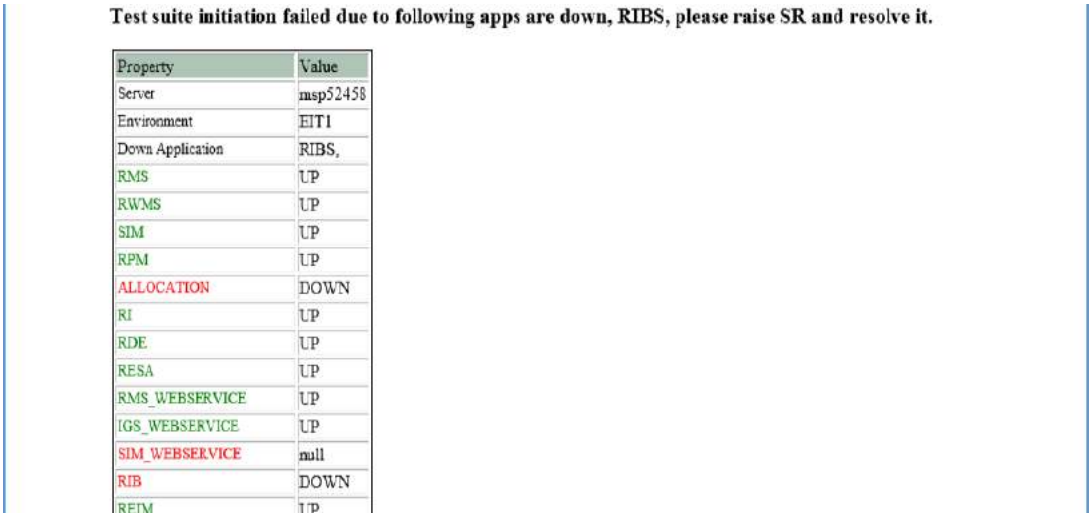
There is a "run.bat" file which is the first program that executes when it comes to automation. The main task of bat file is to facilitate the environment for automation. The very first thing it do is moves the files from gold folder to master folder and then it calls “checkNightlyRun.java” which checks the status of applications. If the status of dependent applications is UP then this bat file calls “Automation_Executor.java” and the run will kick off or else it just sends a mail showing the RIBs are down so nightly run is not initiated.

### 4.2.4 Other Requirements and Constraints

The very first thing that is needed to run status check task is that all the configuration scripts must be placed in the desired production server and the gold folder is having all the required files in it and also the automation executor is place at its desired location. Either update or insert query is executed in each configuration script corresponding to the status of the application in “Automation_config_check” table. “integration_flg” is the main factor that is responsible for application status check task, if its value is set to Y, then only the values will be entered in database and the scheduled check run will start else the nightly run is kicked off directly.

### 4.2.5 Workflow

At the initial stage, before running the automation_executor.java from the production server, the configuration scripts placed in that production server is executed. This configuration script checks the status of dependent applications if the "integration-flg" is set to 'Yes'. Once the configuration script is executed then only the "checkNightlyRun.java" file is executed. This file update the values of status as UP/DOWN in "Automation_config_check" table. Once the table value is populated with UP the nightly run is kicked off else the mail is sent with details of applications whose status are DOWN that RIBs are down and the nightly run cannot be started. Figure 4.6 shows the mail content that is generated with the property and their values and sent when the applications are down.



**Test suite initiation failed due to following apps are down, RIBS, please raise SR and resolve it.**

| Property | Value |
|---|---|
| Server | msp52458 |
| Environment | EIT1 |
| Down Application | RIBS, |
| RMS | UP |
| RWMS | UP |
| SIM | UP |
| RPM | UP |
| ALLOCATION | DOWN |
| RI | UP |
| RDE | UP |
| RESA | UP |
| RMS_WEBSERVICE | UP |
| IGS_WEBSERVICE | UP |
| SIM_WEBSERVICE | null |
| RIB | DOWN |
| REIM | UP |

Figure 4.6 Mail, if some applications are down

## 4.3 EIT Tools and Utility Portal

**Administration:** Administration block contains various utilities like Add User, Add New Connection and Database Connection Check.

**EIT Dashboard:** Graphical representation of analysis of analysis of results and foundation data of various batches is shown in EIT Dashboard.

**EIT Reports:** Tools for generating system options, reports for messages covered, foundation data variables and its comparison report across environments are present in EIT Report.

**EIT Tools:** The functionalities like Configuration management, Data verification, POSU upload generation and Transaction validation are present in EIT Tools.

Figure 4.7 show the UI of the EIT Tools and Utilities Portal.

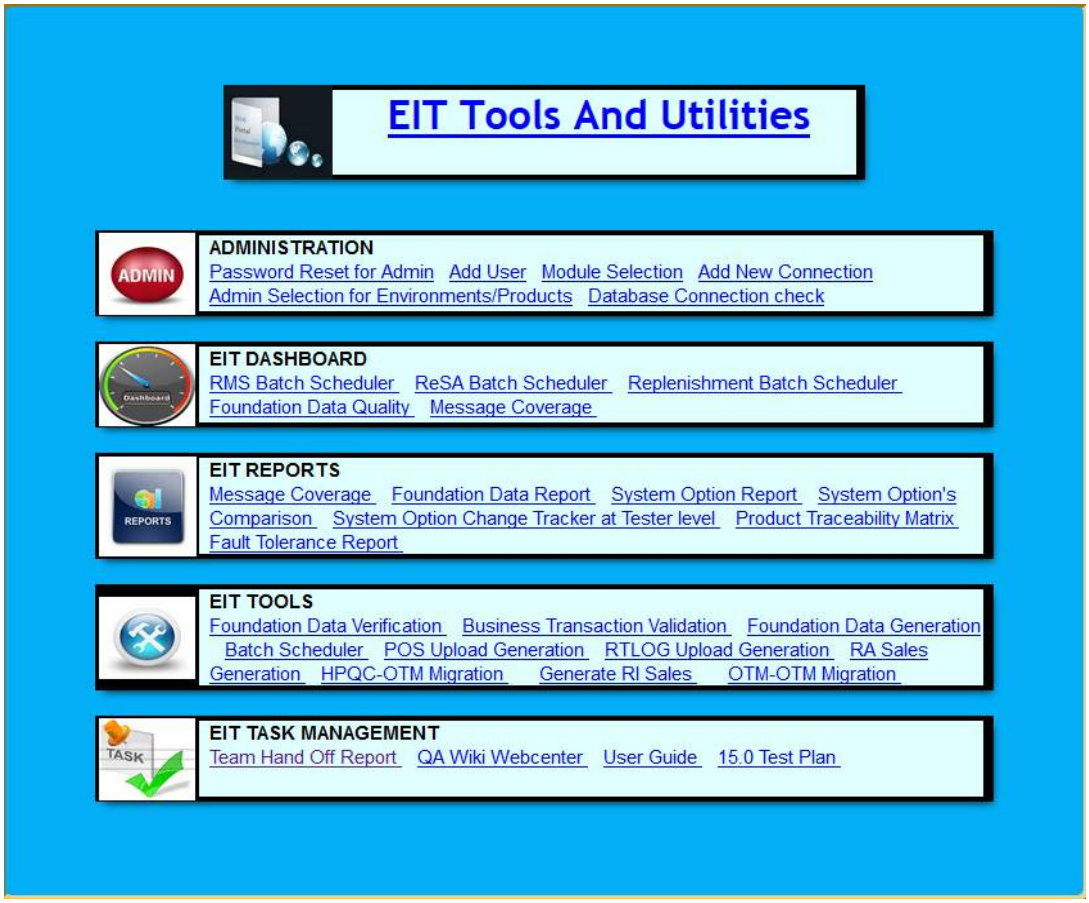



Figure 4.7 EIT Tools and Utilities Portal

### 4.3.1 POS Upload Generation

This utility is used for generating the pre required data for executing the automation scripts for some scenarios like POS Files and X-Store. POS Files and X-Store are used to simulate day to day interaction between the vendor (Shopkeeper) and the customer. The operations under this can be,

- **Sales** – It simulates normal sale of same type of item from the shop.
- **Return** – It simulates normal return of an item to the shop.
- **Exchange** – It simulates normal exchange of an item in the shop i.e. and customer had brought and item in past but now he wants some different item in place of already brought item.
- **Layaway –** It simulates the case where a customer had paid the half amount of the total bill and thus the vendor (shopkeeper) reserves the amount for the particular customer i.e. now logically that quantity is subtracted from the total stock but physically it still exists.
- **Mixed Transaction –** It is combination of Sales and Return in the single transaction.

All of the above files are in .dat format. A simple Sales .dat file example is given in Figure 4.8,

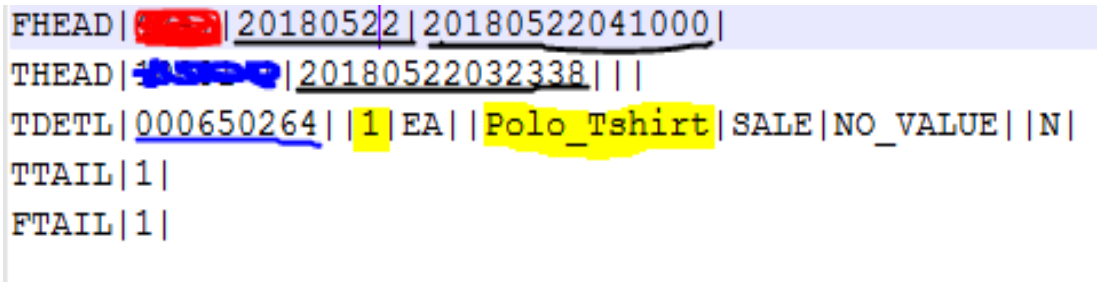



Figure 4.8 Snapshot of simple Sales POS File

The POS File usually contains FHEAD (File Head), THEAD (Table Head), TDETL (Table detail), TTAIL (Table Tail) and FTAIL (File Tail). FHEAD has details about the store for which the transaction is created and also the timestamp. THEAD contains a unique transaction id with which every single transaction can be distinguished. TDETL is the main field in .dat file as it contain the item, quantity as well as the description for the item. TTAIL is the count of how many TDETL are there in a single transaction. There can be many TDETL in a single .dat file.

These files are generated from the utility portal by following the steps in Figure 4.9 and Figure 4.10,


Figure 4.9 POS File Generation

All the required fields are entered for which a POS File has to be created and then it can be simply downloaded from the server.


Figure 4.10 POS File for which .dat file will be created

## 4.4 Continuous Delivery Dashboard

Continuous Delivery Dashboard is used for monitoring and analyzing automation run and its result. Script run results of all the servers can be visualized which helps in determining the failure reason and analyzing the results generated. It reduces the time involved in manual analysis of reports.

### 4.4.1 Dashboard Features

Developed using Express js framework for nodejs for analyzing automation scripts result. Continuous Delivery Dashboard is hosted in Tomcat Server. Dashboard's features are as follow,

- **Dashboard (velocity)** - This page consists of an aggregate of last three months report with Passed, Failed and Total count of the scripts. Data is visualized in bar chart, pie chart and in table format. The snapshot of the report for month of March, April and May with bar graph and pie chart is shown in Figure 4.11.


Figure 4.11 Pie chart representation of Passed, Failed and Total count of scripts

- **Execution Trend Report** - This Page consist of daily run, weekly run and monthly run report. Bar graph and pie chart is generated based on daily, weekly, monthly

number of pass and fail in all servers. The snapshot of the execution trend for month of May with bar graph and pie chart is shown in Figure 4.13.

- **Script Delivery Report** - This Page is to keep track of Script Development Progress. Line graph is generated based on number of planned Scripts versus actual no of Scripts developed. Data is visualized in Line graph and table format and a snapshot can be seen from Figure 4.12.

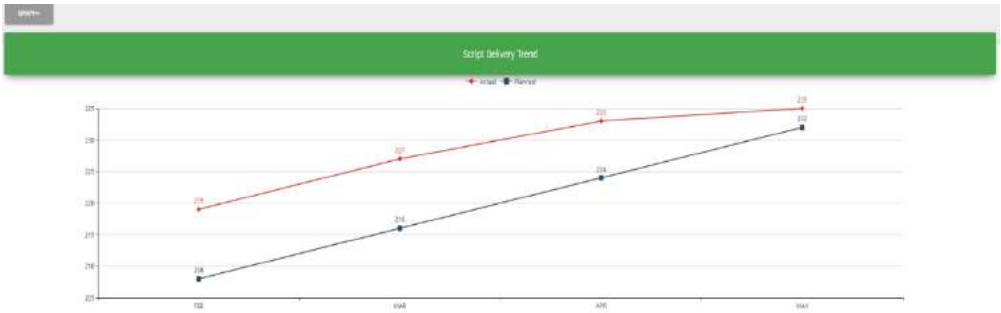



Figure 4.12 Graphical representation of script delivery report


Figure 4.13 Pie chart representation of execution trend

- **Average Execution** - This Page contains all duration of runs in hours and minutes with no of scripts, no of iteration for each server for monthly basis.

# Chapter 5

# Result and Analysis

The observations and findings while working on this project mainly depict why one technology is preferable than the other one or why there was a shift from one technology to another. Also, it shows how one can constantly make changes in existing system to make it more reliable and efficient.

## 5.1 Result of Automation Script

The result can be easily seen from the Automation portal (internal utility which provide one click automation). Multiple scripts can be selected and executed at a time. If single script is selected then the result will appear in the following way with information about Start time, End time, Status of script, Auto Prod in which it was executed, Passed/Failed counts and much more. The Figure 5.1 shows the result for the scenario that we had discussed in section 3.3.1.

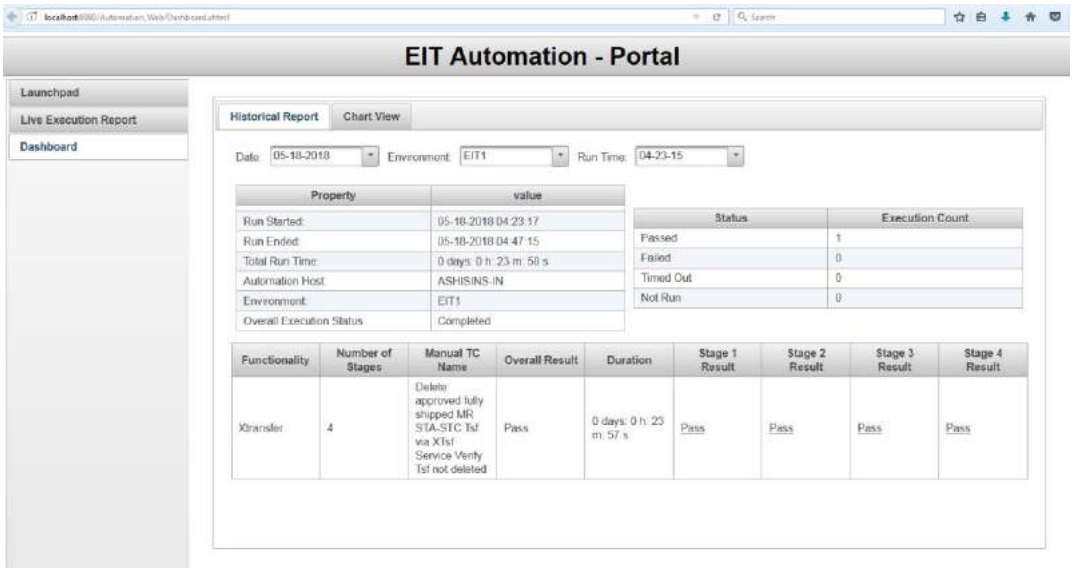



Figure 5.1 Result of the executed script

A detailed result can also be checked by clicking on the Stage Result hyperlink (Pass/Fail). This will show line wise execution result which is very helpful in debugging.

Through the detailed result we can easily know the cause behind the failuer of the scripts. The main reason is found after following the trace of failure i.e. if script is failed in validation part or somewhere while performing desired task then we try to find whats the real reason due to which it got failed as shown in Figure 5.2. There can be many reasons behind this, like if the RIBs are down, Application server itself is down or the desired operation was not performed due to some reasons and thus the validations differ.

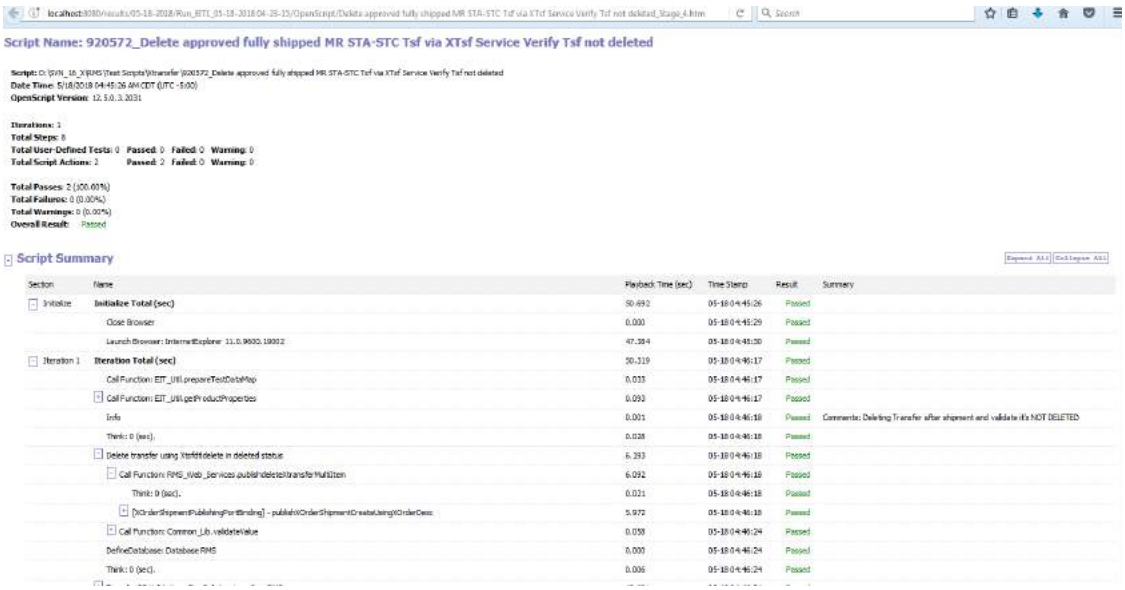



Figure 5.2 Detailed result of executed script

## 5.2 Reasons behind Failure of Scripts

There are several reasons behind the failure of an automation script and most commonly occurring reasons are listed below,

### 5.2.1 Web Page Refreshing

As we work on Cloud environment so for better security the application is defined with several timestamps after which it gets refreshed so as to prevent certain threats. This thing many a times causes an automation script to fail.

### 5.2.2 RIBs

Retail Integration Bus is one of the most commonly occurring cause for scripts failure because a total count of 200+ scripts are running that are distributed into five different Production servers. Some scripts need RIBs to be down and some need the RIBs to be up and working. If at an instance two different scripts on different production server are running who require different state of RIB then one scripts will definitely fail.

### 5.2.3 Server Side Issue

Sometime the server is not responding properly or the Publishers and Subscribers are slow i.e. the data is taking time to flow through RIBs. Due to this also the script gets failed.

### 5.2.4 Application Side Issue (UI)

Sometimes after deployment (new build release) the scripts gets failed. With new advancements in the application some new features are added and old features are modified but the scripts are written according to the older version. Hence they gets failed.

### 5.2.5 Scripting Error

This is also one of the main reason behind the failure. While scripting, if proper testing of script with different scenarios is not done then there might be chance that script is having error (while writing script). Sometime only one condition is tested and it is passed onto the production but his should not be the case.

## 5.3 Contribution of each cause in Failure

When it comes to define the percentage of failure due to one particular cause then one should limit the causes so that a better understanding can be made. In our case, we have limited the causes to the main five reasons (discussed above) behind the failure of scripts. Table 5.1 shows the stats of failure reasons with their respective contribution towards total failure in scripts failure.

Table 5.1 Reasons with their contribution in scripts failure

| Reasons | Total Percentage | Contribution in Failure |
|---|---|---|
| Web Page Refreshing | 100 | 25 % |
| RIBs | 100 | 12 % |
| Server Side Issues | 100 | 24 % |
| Application Side Issues | 100 | 4 % |
| Scripting Error | 100 | 35 % |

These percentage are carefully calculated from the nightly run status by using the following formula,

*Contribution in Failure = Failures caused by particular issue / Total failure count*

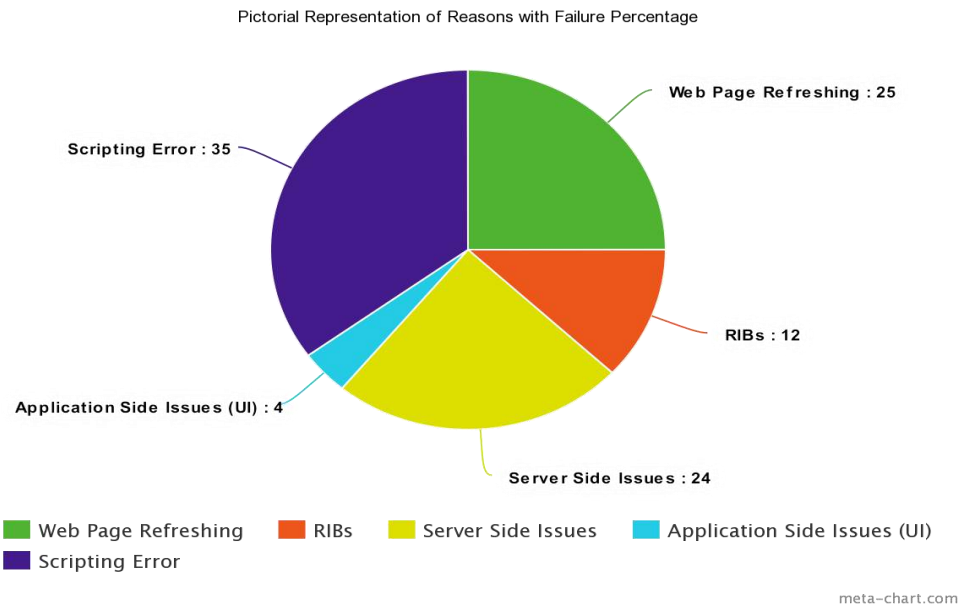



Figure 5.3 Pictorial Representation of reasons with failure percentage

The pictorial representation of failure reasons with their respective contribution towards total failure in scripts failure is shown in Figure 5.3.

## 5.4 Precautions for Reducing Number of Failures

- **Web Page Refreshing** – A proper amount of think time is added in the scripts which prevents the script from failure. As the page is refreshed after certain interval of time so we can add think time accordingly. But more think time leads to long run. Thus it should be added only where ever needed.
- **RIBs –** A proper check is done on all the production servers and then the sequence of execution of scripts in each of the production server is set. This has reduced the RIBs failure to a great extent.
- **Server Side Issues –** Usually this issue arises only when either of the Publishers or Subscriber is down or slow. We can add appropriate think time before moving to another application. Suppose if we are using RMS and SIM then before switching to another application we can add think time to overcome this issues. Sometimes Publishers and Subscribers are forcefully kept down so as to upgrade the application. In this case we can't do anything till the upgradation isn't complete.
- **Application Side Issue (UI) –** This issue logically doesn't have any preventive steps that we can opt for. In this case we must modify existing scripts according to the latest version of application and also keep in mind that previous functionalities should be there.
- **Scripting Error –** These types of error are human error which can only be reduced at the time of scripting.

# Chapter 6

# Conclusion and Future Work

After complete knowledge of the products and tools I worked on the testing of integrated Oracle products. At the later stage, I had to write Android app Testing Scripts using MonkeyTalk and OpenScript. The scripts were successfully tested and sent for production run on nightly basis.

## 6.1 Conclusion

Everyone is worried about the quality of product they are buying and this project helps in attaining a certain level of product quality. Through testing we can know what the limitations of the product are and how they can be overcome. As all the testing is done through automation thus a lot of time is reduced and also the chances of human error. The most occurring reasons that were responsible for degrading product quality are, web page refreshing, RIBs, server side issues, application side issues and scripting error. I would also like to add that automation is a never ending process because with upgradation of application automation might need to be changed.

## 6.2 Future Scope

Machine Learning and Data Mining techniques will be used in future to get better and appropriate reasons for script failure so that a much better software quality can be achieved. EIT Automation Portal will also be enhanced based on the requirement and also for getting deeper into the results of automation scripts. More scenarios of retail world will be covered through automation. All the earlier written scripts will also be kept in mind for consistency in scripts result. Most importantly, regular upgradation of scripts with upgradation of product is necessary else the script failure count will be increased.

# References


[1] Oracle Retail. Oracle functional testing. 2010.

[2] Tanupriya Choudhury Sai Sabitha Jai Gaur, Akshita Goyal. “A walk through of software testing techniques”. *International Conference System Modeling & Advancement in Research Trends (SMART)*, 2016. DOI: 10.1109/SYSMART.2016.7894499

[3] Prasad, Dr. K.V.K.K. “ISTQB Certification Study Guide”, *Wiley*, 2008.

[4] Nagwa Badr Passant Kandil, Sherin Moussa. “A methodology for regression testing reduction and prioritization of agile releases”. *5th International Conference on Information & Communication Technology and Accessibility (ICTA)*, 2015. DOI: 10.1109/ICTA.2015.7426903

[5] Black, Rex. “Managing the Testing Process: Practical Tools and Techniques for Managing Hardware and Software Testing”. *Wiley*, August 2009.

[6] Bernie Gauf Elfriede Dustin, Thom Garrett. “Implementing automated software testing”. *IEEE International Computer Software and Applications Conference*, 2009.

[7] Kolawa, Adam; Huizinga, Dorota. “Automated Defect Prevention: Best Practices in Software Management”. *Wiley-IEEE Computer Society Press*, 2007.

[8] Jingfan Tang. “Towards Automation in Software Test Life Cycle Based on Multi-Agent”. *IEEE*, 2010.

[9] Binder, Robert V. “Testing Object-Oriented Systems: Objects, Patterns, and Tools”. *Addison-Wesley Professional*, 2000. ISBN 978-0321700674.

[10] Claus Klammer and Rudolf Ramler. “A Journey from Manual Testing to Automated Test Generation in an Industry Project”. *IEEE International Conference on Software Quality, Reliability and Security*, 2017.

[11] Manjit Kaur, Raj Kumari, “Comparative Study of Automated Testing Tools: TestComplete and QuickTest Pro”. *International Journal of Computer Applications*, 2011.

[12] Oracle Retail. Oracle retail merchandising system. 2012.

[13] Oracle Retail. Oracle store inventory management. 2013.

[14] Oracle Retail. Oracle warehouse management system. 2013.

[15] Oracle Retail. Oracle retail invoice matching. 2013.

[16] Oracle Retail. Oracle retail price management. 2014.

[17] Oracle Retail. Oracle allocation. 2015.

[18] Oracle Retail. Oracle retail integration bus. 2012.

[19] Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo. "The Oracle Problem in Software Testing". *IEEE transactions on software engineering, vol. 41, no. 5*, may 2015.

[20] Harpreet Kaur, Dr.Gagan Gupta. "Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete". *Int. Journal of Engineering Research and Applications*, 2013. ISSN : 2248-9622.

[21] Jagdish Singh, Monika Sharma. "A Comprehensive Review of Web-based Automation Testing Tools". *International Journal of Innovative Research in Computer and Communication Engineering*, 2015.

[22] K. Valliammai, Dr. P. Sujatha. "Analysis of Efficiency of Automated Software Testing Methods: Direction of Research". *International Journal of Science and Research*.

[23] Vishawjyoti and Sachin Sharma. "Study and analysis of automation testing techniques". *Journal of Global Research in Computer Science*, 2012.

# check6

ORIGINALITY REPORT

| 12% | 9% | 1% | 10% |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | Submitted to Institute of Technology, Nirma University<br>Student Paper | 2% |
| 2 | isa.unomaha.edu<br>Internet Source | 1% |
| 3 | gdeepak.com<br>Internet Source | 1% |
| 4 | Submitted to iGroup<br>Student Paper | 1% |
| 5 | en.wikipedia.org<br>Internet Source | 1% |
| 6 | www.ijera.com<br>Internet Source | <1% |
| 7 | Submitted to Higher Education Commission Pakistan<br>Student Paper | <1% |
| 8 | fr.slideshare.net<br>Internet Source | <1% |

| | | |
|---|---|---|
| 9 | Submitted to NCC Education<br>Student Paper | <1% |
| 10 | Submitted to Asia Pacific University College of Technology and Innovation (UCTI)<br>Student Paper | <1% |
| 11 | www.cs.allegheny.edu<br>Internet Source | <1% |
| 12 | documents.mx<br>Internet Source | <1% |
| 13 | www.reference.com<br>Internet Source | <1% |
| 14 | Submitted to Queen Mary and Westfield College<br>Student Paper | <1% |
| 15 | Submitted to Wawasan Open University<br>Student Paper | <1% |
| 16 | Submitted to ABV-Indian Institute of Information Technology and Management Gwalior<br>Student Paper | <1% |
| 17 | Submitted to London School of Commerce<br>Student Paper | <1% |
| 18 | www.slideshare.net<br>Internet Source | <1% |

| | | |
|---|---|---|
| 19 | Submitted to Whitireia Community Polytechnic<br>Student Paper | <1% |
| 20 | ethesis.nitrkl.ac.in<br>Internet Source | <1% |
| 21 | www.washcost.info<br>Internet Source | <1% |
| 22 | Submitted to Colorado Technical University Online<br>Student Paper | <1% |
| 23 | dar.aucegypt.edu<br>Internet Source | <1% |
| 24 | era.library.ualberta.ca<br>Internet Source | <1% |
| 25 | Submitted to Imperial College of Science, Technology and Medicine<br>Student Paper | <1% |
| 26 | lrd.yahooapis.com<br>Internet Source | <1% |
| 27 | Submitted to Dokuz Eylul Universitesi<br>Student Paper | <1% |
| 28 | digitalcommons.fiu.edu<br>Internet Source | <1% |
| 29 | dspace.thapar.edu:8080<br>Internet Source | <1% |

| | | |
|---|---|---|
| 30 | dspace.mit.edu<br>Internet Source | <1% |
| 31 | jsjobs.org<br>Internet Source | <1% |

| | | | |
|---|---|---|---|
| Exclude quotes | Off | Exclude matches | < 8 words |
| Exclude bibliography | On | | |