Journal of Theoretical and Applied Information Technology
15th July 2018. Vol.96. No 13

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TESTCASE PRIORITIZATION WITH SPECIAL EMPHASIS ON AUTOMATION TESTING USING HYBRID FRAMEWORK

[1]KONERU SRINIVAS, [2]DR. MOHAMMED ISMAIL.B

[1]Research Scholar. Department of Computer Science Engineering, Koneru Lakshmaiah Education. Foundation ,Guntur, Andhra Pradesh, India

[2]Professor. Department of Computer Science Engineering, Koneru Lakshmaiah Education. Foundation, Guntur, Andhra Pradesh, India

E-mail: [1]konerusrinivas254@gmail.com, [2]mdismail@kluniversity.in

ABSTRACT

Testing of the software application is done simultaneously during the software development process, so that defects or errors could be detected at an early stage and any changes made, do not have an adverse effect on the system. Test suite with a different set of test cases is added as a result it keeps growing to a large size. Keeping in mind the resource and time constraints, it is important, implementing test case prioritization, so that core test cases or scripts are executed which are mostly required by the user along with the functionalities or modules that are prone to more bugs. Prioritization techniques will help scheduling test cases for execution, so that faults could be detected at an early stage.

Keywords: *Prioritization Techniques, Automated Tests for Prioritization, Order of Prioritization, Calculating Test Priorities, Categorization of Test Cases, Hybrid framework*

1. INTRODUCTION

Test case prioritization is a method for prioritizing and scheduling the test cases so that test cases with higher priority are scheduled to run first so that cost, effort and time in the software testing phase could be saved. In test case prioritization, the test cases are prioritized and scheduled so that the core functionalities could be tested. Test cases are set with the priority levels based on various factors and the one with highest priority are executed first so that defects could be located at an early stage. Since the testing phase is becoming cumbersome and time consuming, we have proposed various techniques for prioritization of the test cases for automation testing. [1] Choosing an automated tool like Selenium, would be a good choice when test case prioritization is concerned. Selenium IDE is a tool that could be used for Smoke Testing; an overall flow of the system could be recorded and played as and when required. Selenium WebDriver could be used to write test scripts for the various functionalities of the application. TestNG (Test Next Generation) is a powerful and efficient testing framework developed by Cedric Beust with the inspiration from Junit and NUnit that helps in sequencing the tests that needs to run based on the priority required. [2]

2. PROBLEM STATEMENT

Prioritizing the test cases manually proved to be quite complicated, time consuming and became expensive. Automation process in prioritizing test cases proved to be very cost efficient. To leverage the test case prioritization, automation frameworks such as TestNG could be implemented. Hybrid framework approach could be the best solution to maximize the coverage of test cases. Although several studies have already attempted to implement test prioritization techniques but this research work takes to a step ahead in implementing the Automation tool using Hybrid framework, considering cost factor and effectiveness.

SOLUTION TO THE PROBLEM STATED

Novel method of test case prioritization is a very time consuming process and results in very expensive, especially when used with large and complex applications like Banking, Inventory Systems etc. Addressing these challenges, this paper focuses on improving the efficiency by implementing automation tools with a Hybrid Framework. Automation tests prioritizing the test

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

cases based on weightage mentioned in section-5 of CALCULATING TEST PRIORITIES a test suite is developed. This test suite containing Core functionalities of the application are executed first and any changes in the functionalities will allow the regression test to follow thereby running security test, integration test, User interface test, database test etc. utilizing Selenium's TestNG and Hybrid Framework for an efficient result.

3. CATEGORIZATION OF TEST CASES

From a large cluster of test cases, priorities of execution can be decided based upon some rational, non-arbitrary criteria. Prioritization activity is done with an intention to reduce the overall number of test cases in the total testing process. The main objective of the test case prioritization is done to build confidence among the testers and the project leaders that the tests identified for execution are adequate.

Test cases can be categorized into four types. Priority-1 and Priority-2 test cases are the most important as the cost for fixing will be high.

Priority-1: These test cases are responsible for the flow of the software which tests the lifeline of the software. If any of the feature stops working then this will block the further testing and issue has to get fixed on priority. For instance, if the login functionality fails to run then the next screens cannot be reached, so that has to be given top priority.

Priority-2: The test cases that include functionalities required by the end users are tested. These are the functionalities that build confidence in the customer and if it does not work, might result in huge business loss to the software development companies.

Priority-3: In the making of software there would be some unique features to differentiate from competitors. If these stop working, customer may not like but will still use the software when the important features are working fine. Since every customer will look for unique features which might lead to more profits in the business so these stand in the prioritization next to Priority-1 and Priority-2 test cases.

Priority-4: Test cases that stand in this category are related to Cosmetic, User Interface changes or improvements which does not have any impact on using the software. These test cases can be given low priority and sometimes may not be required to execute when there is time shortage in delivering the software. [3]

4. PRIORITISATION TECHNIQUES

Figure-1 shows list of Prioritization techniques which are described as follows:

1. Business Requirements: Based on Business requirements of customers, the test cases that are built to meet the requirements needs to be given the highest priority. There are various factors that could be considered while ranking the priorities of test cases. Test cases which are most important to the customer could be given (CP means Customer assigned Priority), Complex Requirement based test cases are given (CR means Requirement Complexity) and Change Management (CM). A value is assigned to the above mentioned factors and the one having high factor value indicates a need for prioritization of those test cases related to that requirement. The priorities to the test cases are assigned depending upon the priority of the requirement to be tested. The requirement to be tested first is assigned the higher value and the test cases covering that requirement are given higher priority of execution.

2. Coverage-based Heuristics:
   a. Total Statement Coverage Prioritization: This technique is used to prioritize test cases based on the total number of statements covered. And then they are sorted in the order of coverage achieved. If more than one test case, cover the same number of statements then they are randomly ordered.

      Statement Coverage = (Number of Statements Covered / Total No. of Statements) * 100

   b. Additional statement coverage prioritization: Here the test cases are prioritized by incremental number of statements covered.
   c. Total branch coverage prioritization: This technique is same as statement coverage

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

prioritization instead the test cases are prioritized based on the number of branches covered.

Branch Coverage = (Number of Branches Covered / Total No. of Branches) * 100

3. Mutation-based Heuristics:
   a. Total Fault Exposing Potential (TFEP) prioritization: This technique is mostly used to bring out the faults by a test case based on the probability that a fault in that statement might cause a failure for that test case. This will maximize rate of fault detection in a test suite[4].
   b. Additional Fault Exposing Potential (AFEP) prioritization: Here test cases are prioritized based on incremental confidence.
4. Functional Coverage prioritization: This technique is far cheaper than total statement coverage prioritization since the numbers of functions in a program are comparatively less against the number of statements. Here the test cases are prioritized in descending order of the major functions to be executed by the test cases. [5]

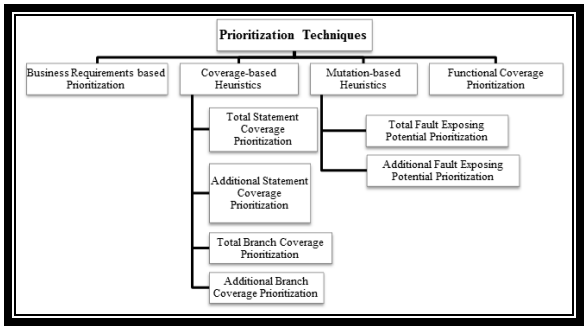

Figure 1: Hierarchical representation of prioritization

5. CALCULATING TEST PRIORITIES

To calculate the test priorities weights must be assigned and then the sum of all the weights are to be calculated and then whichever has the highest weight must be given top priority and so on.

1. Different weights can be allotted in the range from 1 to 10. Considering the values as 1 for low impact, 5 for Medium or Normal impact and 10 for highest impact.
2. Then all the functionalities, crucial requirements of the application must be extracted and points ranging from 1 to 5 needs to be assigned.
3. Calculation of weights for assigning the priorities must be carried out by multiplying the points assigned to the functionality by the relevant weight. Also the impact of the bug must be calculated and then summed up with the value outputted from the functional weights. Table-1 shows the calculation of weights for assigning the priorities. [6]

6. AUTOMATION TECHNIQUE

To Automate the testing process, the right test management tool must be selected and then the test project has to be created where-in all the requirement specification, test scenarios, test cases must be documented. There are many open source test management tools available in the market like Tarantula, Test-link, Jataka Testcube, Mozilla Testopia, Testitool, qaManager, QABook, Radi-testdir, Test Case Web (TCW), Testmaster etc., which automatically synchronizes requirements, test scenarios, defects. These tools can be used to execute automated and manual Test cases. Selenium, Appium, Selendroid, Robotium, cucumber, Watir (Web Application Testing in Ruby), iMacros, Canoo WebTest, SoapUI, KIF (Keep it functional), capybara, Katalon Studio, Windmill, Xmind are some of the open source functional tools that can be used for testing the applications functionality. For load testing some open source tools like Gatling, Apache JMeter, OpenSTA (Open Systems Testing Architecture), Multi-Mechanize etc., can be the best choice. Once the planning of an automated test suite is developed, then prioritisation must be set. Initial tests to be executed would be Smoke test to ensure smooth execution. It should then be followed by Core Functional test, Database test, Security test, Stress test, Interface test and Regression test. Test Cases related to recently modified functions should have a priority. Areas of the product that can cause massive damage should always get more careful testing. The areas of the application that are most used by a typical user should be put under stress testing so that the end users can experience a smooth operation. Techniques like equivalence class partitioning must be implemented. Testing should ensure coverage of all functional areas and if not

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

possible to cover all combinations then at least each should be tried once. Function that consumes most memory must be executed and if that passes, the others are more likely to work. In Normal Case, Executing all the Test cases with Test Coverage is the best practice to full-fledged testing. For effective testing process, test case prioritization and categorization of test cases is very important. Since there is always a time constraint we need to adopt a Risk Based methodology to test case management and execution. Combinatorial Test Design is helpful for the process of Risk Based Testing to keep the test case number explosion under control. There are various tools like Tosca TestSuite, AllPairs, Hexawise etc. These tools use risk-based test design for effective test cases and identify the risk contribution of each test case. They have built-in methodologies to apply techniques such as equivalence partitioning, boundary value analysis and linear expansion for minimizing the number of test cases while increasing risk coverage. These tools when used for test execution, aggregates risk coverage from technical, compliance and business perspective. [7]

6.1 Automated Tests for Prioritization

Figure-2 shows how the automation tests can be prioritized and every modification made to the application a regression test suite could be executed in parallel to trace the bugs. The test suite containing Core functionalities of the application are executed first and any changes in the functionalities will allow the regression test to follow and the same way security test, process flow to check the systems integrity and User interface test to create confidence among the users fraternity which then should be followed by compatibility test. [7-8]

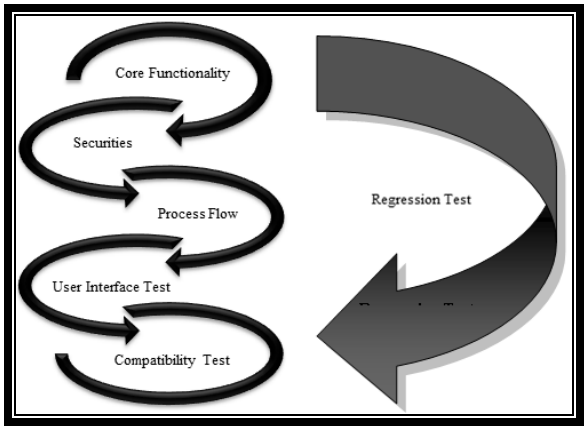

Figure 2: Prioritization of Automation Tests

6.2 Order of Prioritization

1 Core Functionality: Foremost requirement of any client would be their business functionalities, so it becomes the top priority to check whether the core functionalities are working properly.

2 Securities: Next to core functionalities would be the security of the system else the end-user will not be happy to use as there would be threat from hackers losing their passwords and money as well.

3 Process Flow: Then the flow of data from one screen/module to another screen/module must be placed in order.

4 User Interface Test: This phase of testing must come next so that the test cases confirm that Interface objects are in place and receives the right data from the user.

5 Database Test: This test must follow User Interface Testing to check the database is working properly for the various actions such as Add, Update and Delete.

6 Compatibility Test: This test can be placed at the end for checking the cross browser or operating system compatibilities.

7 Data driven Tests: Huge data from a flat file mostly excel is used to send the data into the input fields of the Application which tests the stress. This test is not possible manually as it consumes huge amount of time and lacks accuracy.

8 Compatibility Testing: This test ensures that a wide array of users could operate from different devices, operating systems and browsers. Automation with parallel tests is one of the good options for cross-browser testing.

9 Regression tests: After every modification made by the development team, a re-execution of test suite takes place to ensure possible new bugs are not crept into the system. A test suite can be built with all the above tests to run and whenever changes made in the application, must be ascertained that a new bug is not creeping into the system. [9]

6.3 Prioritizing of test cases based on Time Factor

Considering the tight deadlines, it can further be reduced as follows:

1 Executing the test cases that contain major functionalities.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

2 Parts of the application that are frequently used by the user.
3 Running all the Positive test cases.
4 Performing Stress Test. [8]

6.4 Sample script showing the utilization of testNG

```
importorg.openqa.selenium.By;
importorg.openqa.selenium.WebDriver;
importorg.openqa.selenium.firefox.FirefoxDriv
er;
importorg.testng.Assert;
importorg.testng.annotations.Test;

public class Priority_In_testNG
{
    WebDriver driver;

   // Method 1: Open Firefox Browser
   @Test (priority=1)
   public void openBrowser()
   {
       driver = new FirefoxDriver();
   }

   // Method 2: Launch
https://localhost/OnlineBank

   @Test (priority=2)
   public void launchBankApp()
   {
 driver.get("https://localhost/OnlineBank");
   }

   // Method 3: Verify that the page title is
Login.

   @Test (priority=3)
   public void InitialPageTitleVerification ()
   {
Assert.assertEquals(driver.getTitle().contains("
Login"), true);
      }

   // Method 4: Enter Username and Password

   @Test (priority=4)
   public void processLogin() throws
Exception
  {
     WebElement username =
     driver.findElement(By.id("Unm"));
     username.clear();
     username.sendKeys("Srinivas");
     WebElement password =
     driver.findElement(By.id("Passwd"));
     password.clear();
     password.sendKeys("bank123");
     WebElementSignInButton =
     driver.findElement(By.id("signIn"));
     SignInButton.click();
     Thread.sleep(3000);
  }

   // Method 5: Logout

   @Test (priority=4)
   public void processLogout() throws
Exception
  {
     driver.findElement(By.xpath(".//*[@id='lo
gout']/img")).click();

     driver.close();
   }

}
```

6.5 Selenium Script developed using Hybrid Framework demonstrating Banking Application

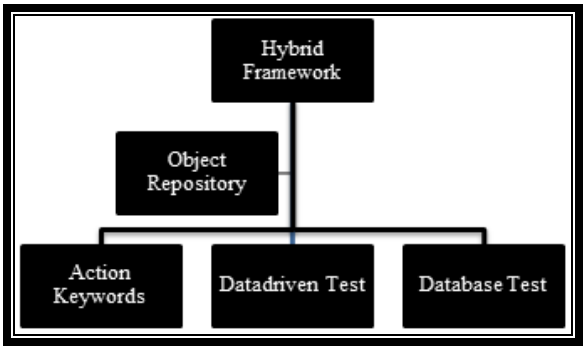

Figure 3: Structure of Hybrid Framework

Hybrid Framework

Banking Application testing using Hybrid framework is shown in figure-3. [9]

Step-1: Creation of repository file with the name bank.properties as follows:

```
URL=https://localhost/OnlineBank
LoginPage.UName=login_id
LoginPage.Pwd=passwrd
LoginPage.signIn=btnsubmit
Logout=.//*[@id='logout']/img
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

menuMaster=.//*[@id='smoothmenu1']/ul/li[5]
/a
menuTrans=.//*[@id='smoothmenu1']/ul/li[5]/
ul/li[1]/a
menuDep=Deposits
linkDep=Make Deposit
txtAcno=account_no
txtAmt=tamt
btnConf=btnConfirm

Step-2: Creation of Excel file for the DATA DRIVEN TEST with the filename "TestData.xlsx" as follows:

	A	B	C
1	Serial-No	Account_Number	Amount
2	1	AB1122789	10000
3	2	XB4357171	560
4	3	BQ4378211	3344
5	4	CB8891023	3456
6	5	BK4422860	7600
7	6	AB6450123	4500
8	7	BB7756289	3200
9	8	XB9044213	8300
10	9	RS7809124	6600
11	10	TP7000212	5000

Step-3: Creation of Excel file with the KEYWORDS in the filename "Keywords.xlsx" as follows:

	A	B	C	D
1	TestCase-ID	Test Step-ID	Description	Action Keyword
2	TC-001	Step-1	Open the Browser	openBrowser
3	TC-001	Step-2	Open the URL "https://localhost/OnlineBank"	openURL
4	TC-001	Step-3	Enter Username	input_Username
5	TC-001	Step-4	Enter Password	input_password
6	TC-001	Step-5	Click Submit	click_Submit
7	TC-001	Step-6	wait for sometime for the page to open	waitFor
8	TC-002	Step-1	Run through the menu to select Masters -> Transactions -> Deposits	goforMenu
9	TC-003	Step-1	Select Make Deposit Link and repeat the data entry inputs from test data file for Account-Numbers & Amounts	dataDriven
10	TC-003	Step-2	Check the database to confirm that correct data is getting saved	check_Database
11	TC-004	Step-1	Click on Logout	click_Logout
12	TC-004	Step-2	Close the Browser	closeBrowser

Step-4: Creation of classfile with the name Constant.java under the package Utility as follows:

```
package Utility;
public class Constant
{
  public static final String q=null;
  public static final String p=null;
  public static final String
dburl="jdbc:mysql://localhost/dbbank";
  public static final String dbuser="root";
  public static final String dbpwd="bank";

  //Excel Test DataSheet with Columns
  public static final int Col_TestSerialNo = 0;

  public static final int Col_Acno = 1 ;
  public static final int Col_Amt = 2;
}
```

Step-5: Creation of class file with the name ExcelUtils.java under the package Utility as follows:

```
package Utility;

importorg.apache.poi.xssf.usermodel.XSSFWorkbo
ok;
importorg.apache.poi.xssf.usermodel.XSSFSheet;
importorg.apache.poi.xssf.usermodel.XSSFCell;
importjava.io.FileInputStream;

public class ExcelUtils
{
private static XSSFWorkbookExcelWBook;
private static XSSFSheetExcelWSheet;
private static XSSFWorkbookDExcelWBook;
private static XSSFSheetDExcelWSheet;

private static int rows;
private static XSSFCell Cell;

public void setExcelFile(String purpose, String
Path, String SheetName) throws Exception
{
FileInputStreamExcelFile=new
FileInputStream(Path);
        if (purpose.equals("datadriven"))
        {
        DExcelWBook = new
        XSSFWorkbook(ExcelFile);
        DExcelWSheet=
        DExcelWBook.getSheet(SheetName);
        }
        else
        {
        ExcelWBook = new
        XSSFWorkbook(ExcelFile);
        ExcelWSheet=
        ExcelWBook.getSheet(SheetName);
        }
}
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
public int getLastRow(String purpose) throws
Exception
{
if (purpose.equals("datadriven"))
rows=DExcelWSheet.getLastRowNum();
else
rows=ExcelWSheet.getLastRowNum();
return rows;
}

public String getCellData(String
purpose,intRowNum, intColNum) throws
Exception
{
if (purpose.equals("datadriven"))
Cell=DExcelWSheet.getRow(RowNum).getCell(C
olNum);
else
Cell=ExcelWSheet.getRow(RowNum).getCell(Col
Num);
String CellData=Cell.getStringCellValue();
returnCellData;
}
}
```

Step-6: Creation of class file with the name BaseClass.java under the package config as follows:

```
packageconfig;
importorg.openqa.selenium.WebDriver;

public class BaseClass
{
public static WebDriver driver;
public  BaseClass(WebDriver driver)
{
        BaseClass.driver = driver;
}
}
```

Step-7: Creation of class file with the name BaseClass.java under the package config as follows:

```
packageconfig;

importjava.io.File;
importjava.io.FileInputStream;
importjava.sql.Connection;
importjava.sql.DriverManager;
importjava.sql.ResultSet;
importjava.sql.SQLException;
importjava.sql.Statement;
importjava.util.Properties;
importjava.util.concurrent.TimeUnit;
importorg.openqa.selenium.By;
importorg.openqa.selenium.WebDriver;
importorg.openqa.selenium.WebElement;
importorg.openqa.selenium.firefox.FirefoxDriver;
importorg.openqa.selenium.interactions.Actions;

importUtility.Constant;
importUtility.ExcelUtils;

public class Action_Keywords extends BaseClass
{
public static Properties prop;

publicAction_Keywords(WebDriver driver)
{
        super(driver);
}

public static void setObjRepository(String Path)
throws Exception
{
        File fn = new File(Path);
        prop = new Properties();
        FileInputStreamobjInput = new
        FileInputStream(fn);
        prop.load(objInput);
        objInput.close();
}

public static void openBrowser()
{
        driver=new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(
        20, TimeUnit.SECONDS);

        driver.manage().window().maximize();
}

public static void openURL() throws
InterruptedException
{

        driver.get(prop.getProperty("URL"));
        driver.manage().timeouts().implicitlyWait(
        20, TimeUnit.SECONDS);
}

public static void input_Username()
{

        driver.findElement(By.id(prop.getProperty
        ("LoginPage.UName"))).sendKeys("admin
        ");
}
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
public static void input_password()
{
        driver.findElement(By.id(prop.getProperty
        ("LoginPage.Pwd"))).sendKeys("123456")
        ;
}

public static void click_Submit()
{
        driver.findElement(By.id(prop.getProperty
        ("LoginPage.signIn"))).click();
}

public static void waitFor() throws Exception
{
        Thread.sleep(5000);
}

public static void goforMenu()
{
        Actions action=new Actions(driver);
        WebElement mas=
        driver.findElement(By.xpath(prop.getProp
        erty("menuMaster")));
        action.moveToElement(mas).build().perfo
rm();
        driver.manage().timeouts().implicitlyWait(
        10, TimeUnit.SECONDS);
        WebElementtrans=
        driver.findElement(By.xpath(prop.getProp
        erty("menuTrans")));
        trans.click();
        driver.manage().timeouts().implicitlyWait(
        10, TimeUnit.SECONDS);
        driver.findElement(By.linkText(prop.getPr
        operty("menuDep"))).click();
}

public static void dataDriven() throws Exception
{
        String sPath =
        "D:\\Selenium_Workspace\\SelProj\\TestD
        ata.xlsx";
        ExcelUtils E2=new ExcelUtils();
        E2.setExcelFile("datadriven",sPath,
        "Sheet1");
        int dataRows =
        E2.getLastRow("datadriven");
        String iAcno;
        String iAmt;

        for(int k=1;k<=dataRows;k++)
        {
                driver.findElement(By.linkText(prop.getPr
                operty("linkDep"))).click();
                driver.manage().timeouts().implicitlyWait(
                10, TimeUnit.SECONDS);

                WebElementAcno=driver.findElement(By
                .id(prop.getProperty("txtAcno")));

                Acno.clear();
                iAcno = E2.getCellData("datadriven", k,
                Constant.Col_Acno);
                Acno.sendKeys(iAcno);
                WebElementAmt=
                driver.findElement(By.id(prop.getProperty
                ("txtAmt")));
                Amt.clear();
                iAmt = E2.getCellData("datadriven", k,
                Constant.Col_Amt);
                Amt.sendKeys(iAmt);
                WebElement sub=
                driver.findElement(By.id(prop.getProperty
                ("btnConf")));
                sub.click();
                check_Database(iAcno,iAmt);
                }
}

public static void check_Database(String iAcno,
String iAmt) throws ClassNotFoundException,
SQLException
{

        Class.forName("com.mysql.jdbc.Driver");
        Connection con =
        DriverManager.getConnection(Constant.d
        burl,
        Constant.dbuser, Constant.dbpwd);
        String query;
        Statement stmt;
        ResultSetrs;
        query="select account_no, amount from
        tbl_CustAccount where
        account_no='"+iAcno+"' and
        amount='"+iAmt+"';";
        stmt=con.createStatement();
        rs=stmt.executeQuery(query);

        while(rs.next())
        {
                if (rs.getString(1) !=null
                &&rs.getString(2)!=null)
                System.out.println("Correct
                record added...");
        }
        rs.close();
```

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

```
        con.close();
}

public static void click_Logout()
{
        driver.findElement(By.xpath(prop.getProp
        erty("Logout"))).click();
}

public static void closeBrowser()
{
        driver.quit();
}
        }
```

Step-8: Creation of class file with the name DriverScript.java under the package ExecutionEngine as follows:

```
packageExecutionEngine;

import org.testng.annotations.Test;
import config.Action_Keywords;
import Utility.ExcelUtils;

public class DriverScript
{
        @Test
        public void Init() throws Exception
        {
        String sPath =
        "D:\\Selenium_Workspace\\SelProj\\Keyw
        ords.xlsx";
        ExcelUtils E1=new ExcelUtils();
        E1.setExcelFile("keyword",sPath,
        "Sheet1");
        int totRows=E1.getLastRow("keyword");
        String
        objPath="D:\\Selenium_Workspace\\SelPr
        oj\\src\\JCC.properties";

        Action_Keywords.setObjRepository(objPa
th);
        for (int iRow=1;iRow<=totRows;iRow++)
        {
        String sActionKeyword =
E1.getCellData("keyword",iRow, 3);
if(sActionKeyword.equals("openBrowser"))
        Action_Keywords.openBrowser();
else if(sActionKeyword.equals("openURL"))
        Action_Keywords.openURL();
else if(sActionKeyword.equals("input_Username"))
        Action_Keywords.input_Username();
else if(sActionKeyword.equals("input_password"))
        Action_Keywords.input_password();
else if(sActionKeyword.equals("click_Submit"))
        Action_Keywords.click_Submit();
else if(sActionKeyword.equals("waitFor"))
        Action_Keywords.waitFor();
else if(sActionKeyword.equals("goforMenu"))
        Action_Keywords.goforMenu();
else if(sActionKeyword.equals("dataDriven"))
        Action_Keywords.dataDriven();
else if(sActionKeyword.equals("click_Logout"))
        Action_Keywords.click_Logout();
else if(sActionKeyword.equals("closeBrowser"))
        Action_Keywords.closeBrowser();
                }
        }
}
```

Above script when executed, reads the keywords from Keywords.xlsx file and then branches the control into the package config and into the class Action_Keywords and runs every test case. It also gets the objects defined in the object repository file to locate the objects in the application and also reads the data from testdata.xlsx file to send the data as inputs into the application. This script is the best example for Hybrid framework as it contains a combination of various frameworks Keyword Driven, Data Driven, Functionalities broken into multiple parts and also uses an object repository file.

7. DIFFERENCE FROM PRIOR WORK

In my earlier paper suggested the *TEST AUTOMATION PLAN FOR FINANCIAL APPLICATIONS* and focused mainly on utilization of Hybrid Framework. This paper gives the solution for Hybrid Framework with respect to Test Prioritization.

8. LIMITATIONS

The limitation with this approach is that there would be a long time between development and software releases and because the two teams work separately, the development team will not be aware of operational roadblocks that might prevent the program from working as anticipated. My research in future work is to focus on best practices to meld application development and deployment into a more streamlined process that aligns development, quality assurance (QA) and operations team efforts.

9. CONCLUSION

If the core practices of Risk Management and Risk Based Testing are implemented then it can help in bringing out a better system. When automated

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

testing is applied, expansion of test suite is no more a risk and test coverage gets maximized which then permits prioritization of low severity test cases.

In summary, this research demonstrates the test case prioritization techniques and its effective implementation with Automation Tools using TestNG Framework for speeding up the testing process a step further.

ACKNOWLEDGEMENT

I thank Dr. Mohammed Ismail, Professor, Computer Science Engineering Department, K.L.E.F University, Vaddeswaram, Vijayawada, Andhra Pradesh, for his valuable guidance and extensive support in completing this paper.

REFRENCES:

[1] Elbaum, S., A. Malishevsky and G. Rothermel, 2002. Test case prioritization: A family of empirical studies. IEEE Trans. Software Eng., 28: 159182.

[2] Matthias Ratert, Secusmart GmbH. “Automated, tool independent test case prioritization”, EuroStar Software Testing Conference, Germany. Year 2012.

[3] Joachim Karlsson & Kevin Ryan, "A Cost-Value Approach for Prioritizing Requirements", IEEE Software, Sept. 1997.

[4] Mohammed Ismail. B, B. Eswara Reddy, T. Bhaskara Reddy “Cuckoo Inspired Fast Search Algorithm for Fractal Image Encoding” Elsevier Journal of King Saud University Computer and Information Sciences November 2016

[5] G. Rothermel ; R.H. Untch ; Chengyun Chu ; M.J. Harrold. Test case prioritization: an empirical study, IEEE 6 August 2002.

[6] Andreas Hoffmann, Axel Rennoch, Ina Schieferdecker and Nicole Radziwill, "A Generic Approach for Modeling Test Case Priorities with Applications for Test Development and Execution", 2009 - Research Gate, Germany.

[7] Graves, T.L., M.J. Harrold, J.M. Kim, Ad. Porter and G. Rothermel, 2001. An empirical study of regression test selection techniques. ACM Trans. Software Eng. Methodol., 10: 184208.

[8] Dr.Mohammed Ismail, C.H.S.L. Sowmya, T. Sai Sudheera ,P Ravi Teja “A Comparative Study on Dealing with Sparsity in E-Commerce" International Journal of Pure and Applied Mathematics Volume 118 No. 5 Jan 2018, 185-194

[9] "https://crossbrowsertesting.com/blog/test-automation/prioritizing-test-automation", "Prioritizing Tests When You Can’t Automate Everything", May 25, 2017

[10] Malishevsky, A.G., J.R. Ruthruff, G. Rothermel and S. Elbaum, 2006. Cost-cognizant test case prioritization. Technical Report TR-UNL-CSE-2006-0004, Department of Computer Science and Engineering, University of Nebraska-Lincoln. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.112.9150.

[11] Bhagyashree Bhondokar, Pooja Ranawade, Snehal Jadhav and Mayuri Vibhute, “Hybrid Test Automation Framework for Web Application”, International Journal of Engineering Research & Technology, Vol. 4 - Issue 04 (April - 2015)

[12] “http://toolsqa.com/selenium-webdriver/selenium-automation-hybrid-framework”, 5 October 2017

.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1: Calculation of Test Priorities

Functionalities	Critical Features	Application Interface	Complexity	Change Frequency	Risk Calculation
Weights	**3**	**10**	**3**	**3**	
Customer Account Creation	5	2	3	2	35*15 = 525
Administrator Services	4	2	4	2	32*18 = 576
Banking Services	2	3	3	3	36*18 = 648
Transactions	4	5	2	4	62*18 = 1116
Reports	5	4	1	1	55*6 = 330

Risk Calculation = [(Weights * Critical Features) + (Weights * Application Interface)] * [(Weights * Complexity) + (Weights * Change Frequency)]

Customer Account Creation = [(3 * 5) + (10 * 2)] * [(3 * 3) + (3 * 2)]
= [15 + 20] * [9 + 6]
= [35] * [15]
= 525