Sri Indu College of Engneering and technology

SOFTWARE TESTING METHODOLOGIES LAB

DEPARTMENT OF CSE

INDEX

Week.No	Experiment	DATES	MARKS	SIGN
1.	CONSTRUCTS			
2.	INTROSPECTION OF MATRIX MULTIPLICATION			
3.	SYSTEM SPECIFICATIONS			
4.	TEST CASES			
5.	TEST PLAN			
6.	TESTING TOOL			
7.	SELENIUM			
8.	BUGBIT (TEST MANAGEMENT TOOL)			
9.	BUG TRACKING TOOL (BUGZILLA)			
10.	OPEN SOURCE TESTING TOOL			

EXPERIMENT # 1

CONSTRUCTS

1.1 OBJECTIVE:

Write a C program to demonstrate the working of the following constructs:

i. do…while
ii. while…do
iii. if …else
iv. switch
v. for Loops.

2.1 PROGRAM LOGIC:

1. do …while

declare i,
intialize n to 5 and j to 0.
read i value .
loop :
if (i%2== 0)
print i as even number. And
increment i and j value.
otherwise print i as odd number . and increment i and j value .
if i>0 and j<n go to loop

2. while

declare i, intialize n to 5 and j to 0. read i value .
loop : if i>0 and j<n.
if i%2 == 0 print i as even number and increment i and j value.
otherwise print i as odd number and increment i and j value .
go to loop

3. if….else

declare i value. read i value .
if i%2== 0 print i as even number.
otherwise odd number.

4. switch

declare a,b,c.
read i value.
print enter a,b values . read a,b values . switch (case value = i)
if case value = 1
c is sum of a and b
if case value =2
c is difference of a and b
if case value = 3
c is multiplication of a and b. if case value = 4 c is division of a and b .

5. for loop

declare i value.
read i value .
loop initialize i to 1 if i<=5
print i as even number
else
print as odd number. increment i value

1.4 PROCEDURE:

1. Create : Open editor vi x.c write a program after that press ESC and: wq for save and Quit.
2. Compile: gcc x.c.
3. Execute: . / a.out.

1.5 SOURCE CODE:

1. do…while

```
#include
<stdio.h>
 void main ()
{
int i, n=5,j=0;
 printf("enter a no");
 scanf("%d",&i);
do
{ if(i%2==
0)
{
```

```
printf("%d", i);
printf("is a even no.");
i++;
j++;
}
else
{
printf("%d", i);
printf("is a odd no.\n"); i++; j++;
}
}
while(i>0&&j<n);
getch();
```

2. while

```
#include<stdio.h>
#include <conio.h>
void main ()
{
int i,n=5,j=1;
printf(—enter a no‖);
scanf("%d",&i);
while (i>0 && j<n)
{ if(i%2==
0)
{
printf("%d",i);
printf("is a even number");
i++;
j++;
}
else
{
printf("%d",i);
printf("its a odd number");
 i++;
 j++;
}
}
getch();
```

3. if….else

```
 #include<stdio.h>
#include <conio.h>
void main ()
{
int I,c;
printf(“enter a number “);
 scanf(“%d” ,&i);
 if(i%2==0)
{
printf(“%d” ,i);
printf({“s a even number”);
}
else
{
printf(“%d”,i);
printf(“is a odd number”);
}
}
```

4. switch

```
#include<stdio.h>
#include <conio.h>
void main()
{
int a,b,c;
printf(“1.add/n 2.sub /n 3.mul /n 4.div /n enter your choice”);
scanf(“%d”, &i);
printf(“enter a,b values”);
scanf(“%d%d” ,&a,&b);
switch(i)
{
case 1: c=a+b;
printf(“the sum of a & b is: %d”,c);
break;
case 2: c=a-b;
printf(”the diff of a & b is: %d” ,c);
break;
 case 3: c=a*b;
```

```
printf("the mul of a & b is: %d",c);
break;
case 4: c=a/b;
printf("the div of a & b is: %d ,c);
break:;
default:
printf("enter your choice");
 break;
}
getch();
}
```

5. for

```
#include<stdio.h>
#include <conio.h>
main()
{
int i;
printf("enter a no");
scanf("%d",&i); for(i=1;i<=5;i++)
{ if(i%2==
0)
{
printf("%d", i);
printf("is a even no");
i++;
}
else
{
printf("%d", i);
 printf("is a odd no");
 i++;
}
}
getch();
}
```

1.6 INPUT/OUTPUT

1. do…while

INPUT	ACTUAL OUTPUT
2	2 is even number 3 is odd number 4 is even number 5 is odd number 6 is even number

Test cases:

Test case no: 1
Test case name: Positive values within range

Input	Expected output	Actual output	Remarks
2	2 is even number 3 is odd number 4 is even number 5 is odd number 6 is even number	2 is even number 3 is odd number 4 is even number 5 is odd number 6 is even number	Success

Test case no: 2
Test case name: Negative values within a range

Input	Expected output	Actual output	Remarks
2	-2 is even number -3 is odd number -4 is even number -5 is odd number -6 is even number	-2 is an even number	fail

Test case no: 3
Test case name: Out of range values testing

Input	Expected output	Actual output	Remarks
123456789122222222222	1234567891222222222213	234567891222222215	fail

2. while

Input	Actual output
2	2 is even number 3 is odd number 4 is even number 5 is odd number 6 is even number

Test cases: Test case no: 1
Test case name: **Positive values within range**

Input	Expected output	Actual output	Remarks
2	2 is even number 3 is odd number 4 is even number 5 is odd number 6 is even number	2 is even number 3 is odd number 4 is even number 5 is odd number 6 is even number	success

Test case no:2
Test case name: Negative values within a range

Input	Expected output	Actual output	Remarks
-2	-2 is even number -3 is odd number -4 is even number -5 is odd number -6 is even number	-2 is an even number	fail

Test case no: 3
Test case name: Out of range values testing

Input	Expected output	Actual output	Remarks
123456789122222222222	1234567891222222222213	234567891222222215	fail

3. if …else

Input	Actual output
2	2 is even number 3 is odd number 4 is even number 5 is odd number 6 is even number

Test cases:
Test case no: 1

Test case name: Positive values within range

Input	Expected output	Actual output	Remarks
2	2 is even number	2 is even number	success
	3 is odd number	3 is odd number	
	4 is even number	4 is even number	
	5 is odd number	5 is odd number	
	6 is even number	6 is even number	

Test case no:2

Test case name: Negative values within a range.

Input	Expected output	Actual output	Remarks
-2	-2 is even number -3 is odd number -4 is even number -5 is odd number -6 is even number	-2 is an even number	fail

Test case no: 3

Test case name: Out of range values testing

Input	Expected output	Actual output	Remarks
123456789122222222222	12345678912222222213	234567891222222215	fail

4. switch

Input	Actual output
Enter Ur choice: 1 Enter a, b Values: 3, 2	The sum of a & b is:5
Enter Ur choice: 2 Enter a, b Values: 3, 2	The diff of a & b is: 1
Enter Ur choice: 3 Enter a, b Values: 3, 2	The Mul of a & b is: 6
Enter Ur choice: 4 Enter a, b Values: 3, 2	The Div of a & b is: 1

1.7 PRE LAB VIVA QUESTIONS:

1. What are different loop statements in C?
2. Compare entry controlled and exit controlled loops?
3. What is the use of break statement?
4. Compare different if statements in C?
5. State different data types and ranges in C?

EXPERIMENT # 2

INTROSPECTION OF MATRIX MULTIPLICATION

2.1 OBJECTIVE:

A program written in c language for matrix multiplication fails -Introspect the causes for its failure and write down the possible reasons for its failure.

2.2 PROGRAM LOGIC:

1. Read the no. of rows (r1) and cols (c1) of a matrix a[3][3].
2. Read the no. of rows (r2) and cols. (c2) of matrix b[3][3].
3. If c1=r2 then display matrix multiplication is possible otherwise display impossible
4. If c1=r2 then read the elements into both the matrices a andb.
5. Initialize a resultant matrix c[3][3] with 0.
6. Calculate c[i][j] = c[i][j] + a[i][k] * b[k][j].
7. Display the resultant matrix

2.3 PROCEDURE:

a. Create : Open editor vi x.c write a program after that press ESC and: wq for save and Quit.
b. Compile: gcc x.c.
c. Execute: . / a.out.

2.4 SOURCE CODE :

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[3][3],b[3][3],c[3][3],i,j,k,m,n,p,q;
clrscr();
 int a[3][3],b[3][3],c[3][3],i,j,k,m,n,p,q;
clrscr();
printf(—Enter matrix no.of rows & cols); scanf(—%d%d‖,&m,&n);
printf(— Enter 2matrix no.of rows & cols‖) ;
scanf(—%d%d‖,&p,&q);
```

```
printf("\n enter the matrix elements"); for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
scanf("%d",&a[i][j]);
}
}
printf("\n a matrix is\n"); for(i=0;i<m;i++)
{
for(j=0;j<n;j++)
{
printf("%d\t",a[i][j]);
}
printf("\n");
}
for(i=0;i<p;i++)
{
for(j=0;j<q;j++)
{
scanf("%d\t",&b[i][j]);
}
}
printf("\n b matrix is\n");
for(i=0;i<p;i++)
{
for(j=0;j<q;j++)
{
printf("%d\t",b[i][j]);
}
printf("\n");
}
for(i=0;i<m;i++)
{
for(j=0;j<q;j++)
{
c[i][j]=0; for(k=0;k<n;k++)
{
c[i][j]=c[i][j]+a[i][k]*b[k][j];
}
}
}
for(i=0;i<m;i++)
{
```

```
for(j=0;j<q;j++)
{
printf("%d\t",c[i][j]);
}
printf("\n");
}
getch();
}
```

2.5 INPUT AND OUTPUT

Input	Actual Output
Matrix1: 1 1 1 1 1 1 1 1 1 Matrix2: 1 1 1 1 1 1 1 1 1	3 3 3 3 3 3 3 3 3

Test cases:

Test case no: 1

Test case name: Equal no. of rows & cols

Input	Expected output	Actual output	Remarks
Matrix1 rows & cols= 3 3 Matrix2 rows & cols= 3 3 Matrix1: 1 1 1 1 1 1 1 1 1 Matrix2: 1 1 1 1 1 1 1 1 1	3 3 3 3 3 3 3 3 3	3 3 3 3 3 3 3 3 3	Success

Test case no: 2

Test case name: Cols of 1st matrix is not equal to rows of 2nd matrix.

Input	Expected output	Actual output	Remarks
Matrix1 rows & cols= 2 2 Matrix2 rows & cols= 3 2	Operation can‘t be performed		fail

Input	Expected output	Actual output	Remarks
Matrix1 rows & cols= 2 2 Matrix2 rows & cols= 2 2 1234567891 2222222222 2234567891 2222222221 234567891 22222221533 213242424 56456475457			fail

2.6 VIVA QUESTIONS:

1. What is an array?
2. State 2-dimensional and multi-dimensional arraysyntax?
3. Difference between array and structure?
4. What is a structure and specify its syntax?
5. Write syntax for multidimensional arrays?

EXPERIMENT # 3

SYSTEM SPECIFICATIONS

3.1 OBJECTIVE:

1. Study the system specifications of ATM system and report various bugs in it.
2. Study the system specifications of banking application and report various bugs in it.

3.2 BUGS IN ATM SYSTEM:

- Machine is accepting ATM card.
- Machine is rejecting expired card.
- Successful entry of PIN number.
- Unsuccessful operation due to enter wrong PIN number 3 times.
- Successful selection of language.
- Successful selection of account type.
- Unsuccessful operation due to invalid account type.
- Successful selection of amount to be withdrawn.
- Successful withdrawal.
- Expected message due to amount is greater than day limit.
- Unsuccessful withdraw operation due to lack of money in ATM.
- Expected message due to amount to withdraw is greater than possiblebalance.
- Unsuccessful withdraw operation due to click cancel after insert card.

2.2 VIVA QUESTIONS:

1. What are design specifications?
2. What are different types of bugs?
3. .Compare functional and structural testing?
4. Explain dichotomies of testing?
4. What are consequences of bugs?

EXPERIMENT # 4

TEST CASES

4.1 OBJECTIVE:

Study the Test Cases of banking applications and report the various bugs in it.

4.2 TEST CASES FOR BANKING APPLICATION:

1. Checking mandatory input parameters.
2. Checking optional input parameters.
3. Check whether able to create account entity.
4. Check whether you are able to deposit an amount in the newly created account (andthus updating the balance).
5. Check whether you are able to withdraw an amount in the newly created account(after deposit) (and thus updating the balance).
6. Check whether company name and its pan number and other details are provided in case of salary account.
7. Check whether primary account number is provided in case of secondary account.
8. Check whether company details are provided in cases of company's current account.
9. Check whether proofs for joint account are provided in case of joint account.
10. Check whether you are able deposit an account in the name of either of the person in a joint account.
11. Check whether you are able withdraws an account in the name of either of the person in a joint account.
12. Check whether you are able to maintain zero balance in salary account.
13. Check whether you are not able to maintain zero balance (or mini balance) in non-salary account.

4.3 VIVA QUESTIONS:

- What is flow graph testing?
- Difference between flow graph and control graph?
- Compare data and coding bugs?
- Differentiate testing and debugging?
- Explain complexity barrier

EXPERIMENT # 5

TEST PLAN

5.1 OBJECTIVE:

Create a test plan document for any application (e.g. Library Management System)

5.2 TEST PLAN DOCUMENT FOR LIBRARY MANAGEMENT SYSTEM:

The Library Management System is an online application for assisting a librarian imagining book library in a University. The system would provide basic set of features to add/update clients, add/update books, search for books, and manage check-in / checkout processes. Our test group tested the system based on the requirement specification .This test report is the result for testing in the LMS. It mainly focuses on two problems

1. What we will test
2. How we will test.

1. GUI TEST

Pass criteria: librarians could use this GUI to interface with the backend library database without any difficulties.

2. DATABASE TEST

Pass criteria: Results of all basic and advanced operations are normal (refer to section 4)

3. BASIC FUNCTION TEST ADD A STUDENT

1. Each customer/student should have following attributes: Student ID/SSN (unique), Name, Address and Phone number.

2. The retrieved customer information by viewing customer detail should contain the four at- tributes.

3. UPDATE/DELETE STUDENT

1. The record would be selected using the student ID.

2. Updates can be made on full. Items only: Name, Address, Phone number .The record can be deleted if there are no books issued by user. The updated values would be reflected if the same customer's ID/SSN is called for.

3. CHECK-IN BOOK

1. Librarians can check in a book using its call number

2. The check-in can be initiated from a previous search operation where user has selected a set of books.
3. The return date would automatically reflect the current system date.

4. Any late fees would be computed as difference between due date and return date at rate of 10 cents a day.

5.3 VIVA QUESTIONS:

1. Difference between domain and path testing?
2. What is testing blindness?
3. What is path instrumentation?
4. What are graph matrices?
5. Define slice and dice?

EXPERIMENT # 6

TESTING TOOL

6.1 OBJECTIVE :

Study of Any Testing Tool(Win Runner)

6.2 STUDY OF WIN RUNNER TESTING TOOL :

1. Win Runner is a program that is responsible for the automated testing of software.
2. Win Runner is a Mercury Interactive enterprise functional testing tool for Microsoft windows applications.

Importance Of Automated Testing:

- Reduced testing time Consistent test procedures
- Reduces QA cost
- Improved testing productivity

Win Runner Uses :

- ✓ Win Runner *uses* TSL, or Test Script Language. The goal of Win Runner is to make sure business processes are properly carried out.

- ✓ Another impressive aspect of Win Runner is the ability to record various interactions, and transform them into scripts. Win Runner is designed for testing graphical user interfaces. When the user make an interaction with the GUI, this interaction can be recorded. Re-cording the interactions allows determining various bugs that need to be fixed. When the test is completed, Win Runner will provide with detailed information regarding the results.

Win Runner Testing Modes

1. Context Sensitive

Context Sensitive mode records your actions on the application being tested in terms of the GUI objects you select (such as windows, lists, and buttons), while ignoring the physical location of the object on the screen. Every time you perform an operation on the application being tested, a

TSL statement describing the object selected and the action performed is generated in the test script. As you record, Win Runner writes a unique description of each selected object to a GUI map.

2. Analog

Analog mode records mouse clicks, keyboard input, and the exact x and y coordinates traveled by the mouse. When the test is run, Win Runner retraces the mouse tracks. Use Analog mode when exact mouse coordinates are important to your test, such as when testing a drawing application.

USING WIN RUNNER WINDOW

Before you begin creating tests, you should familiarize yourself with the Win Runner main window.

To start Win Runner:

Choose Programs>Win Runner>Win Runner on the Start menu.

The first time you start Win Runner, the Welcome to Win Runner window and the What_s New in Win Runner help open. From the Welcome window you can create a new test, open an existing test, or view an overview of Win Runner in your default browser. If you do not want this window to appear the next time you start Win Runner, clear the **Show on Startup** check box.To show the **Welcome to Win Runner** window upon startup from within Win Runner, choose **Settings > General Options**, click the **Environment** tab, and select the **Show Welcome screen** checkbox.

The Main Win Runner Window

The main Win Runner window contains the following key elements:

- ✧ Win Runner title bar
- ✧ Menu bar, with drop-down menus of Win Runner commands
- ✧ Standard toolbar, with buttons of commands commonly used when running a test
- ✧ User toolbar, with commands commonly used while creating a test
- ✧ Status bar, with information on the current command, the line number of the insertion point and the name of the current results folder
- ✧ The Standard toolbar provides easy access to frequently performed tasks, such as opening, executing, and saving tests, and viewing test results.

EXPERIMENT # 6(A)

Create a script by recording in **Context Sensitive mode** that tests the process of opening an order in the Flight Reservation application. You will create the script

1. **Start Win Runner.**

   If Win Runner is not already open, choose Programs > Win Runner > Win Runner on the Start menu.

2. **Open a new test.**

   If the Welcome window is open, click the New Test button. Otherwise, choose File > New.

   A new test window opens in Win Runner.

3. **Start the Flight Reservation application and log in.**

   Choose Programs > Win Runner > Sample Applications > Flight 1A on the Start menu. In the Login window, type your name and the password mercury, and click OK. The name you type must be at least four characters long. Position the Flight Reservation application and Win Runner so that they are both clearly visible on your desktop.

4. **Start recording in Context Sensitive mode.**

   In WinRunner, choose Create > Record—Context Sensitive orclick the Record button on the toolbar. From this point on, Win Runnerrecords all mouse clicks and keyboard input. Note that the text, —Rec‖ appears in blue above the recording button. This indicates that you are recording in Context Sensitive mode. The status bar also informs you of your current recording mode.

5. **Open order #3.**

   In the Flight Reservation application, choose File > Open Order. In the Open Order dialog box, select the Order No. check box. Type 3 in the adjacent box, and click OK. Watch how Win Runner generates a test script in the test window as you work.

6. **Stop recording.**

   In Win Runner, choose Create > Stop Recording or click the Stop button on the toolbar.

7. **Save the test.**

   Choose File > Save or click the Save button on the toolbar. Save the test as lesson3 in a convenient location on your hard drive. Click Save to close the Save Test dialog box. Note that Win Runner saves the lesson3 test in the file system as a folder, and not as an individual file. This folder contains the test script and the results that are generated when you run the test.

   Output: Win Runner Test Results window is open and displays the test results.

   Conclusion: Recording in Context Sensitive mode is cleared and test results are also seen.

EXPERIMENT # 6(B)

Aim: Purpose of this exercise is to Study Synchronizing test

Synchronizing test

When you run tests, your application may not always respond to input with the same speed. For example, it might take a few seconds:

1. To retrieve information from a database
2. For a window to pop up
3. For a progress bar to reach 100%
4. For a status message to appear

Input: Creating a Test

In this first exercise you will create a test that opens a new order in the Flight Reservation application and inserts the order into a database.

1. Start Win Runner and open a new test.

If Win Runner is not already open, choose Programs > Win Runner > Win Runner on the Start menu. If the Welcome window is open, click the New Test button. Otherwise, choose File > New. A new test window opens.

2. Start the Flight Reservation application and log in.

Choose Programs > Win Runner > Sample Applications > Flight 1A on the Start menu. In the Login window, type your name and the password mercury, and click OK. Reposition the Flight Reservation application and Win Runner so that they are both clearly visible on your desktop.

3. Start recording in Context Sensitive mode.

Choose Create > Record Context Sensitive or click the Record button on the tool bar.Win Runner will start recording the test.

4. Create a new order.

Choose File > New Order in the Flight Reservation application.

5. Fill in flight and passenger information.

6. Insert the order into the database.

Click the Insert Order button. When the insertion is complete, the -Insert Done message appears in the status bar.

7. Delete the order.

Click the Delete Order button and click Yes in the message window to confirm the deletion.

8. Stop recording.

Choose Create > Stop Recording or click the Stop button.

9. Save the test.

Choose File > Save. Save the test as lesson4 in a convenient location on your hard drive. Click Save to close the Save Test dialog box.

Output: Win Runner Test Results window is open and displays the test results.

Conclusion: Importance of Synchronizing test is cleared and test results are also seen.

VIVA QUESTIONS:

1. Define win runner?
2. Explain uses of win runner?
3. What are different modes of win runner?
4. Explain win runner testing process?
5. How to test a module using win runner?
6. Create a script in win runner in context sensitive mode?
7. What are the steps for synchronizing test in win runner?

EXPERIMENT # 7

SELENIUM

7.1 OBJECTIVE:

Study of any web testing tool (e.g. Selenium)

7.2 STUDY OF SELENIUM WEB TESTING TOOL:

1. Selenium is a robust set of tools that supports rapid development of test automation for web-based applications. Selenium provides a rich set of testing functions specifically geared to the needs of testing of a web application. These operations are highly flexible, allowing many options for locating UI elements and comparing expected test results against actual application behavior.
2. One of Selenium_s key features is the support for executing one_s tests on multiple browser platforms.
3. Selenium Components
4. Selenium is composed of three major tools. Each one has a specific role in aiding the development of web application test automation.

Selenium-RC provides an API (Application Programming Interface) and library for each of its supported languages: HTML, Java, C#, Perl, PHP, Python, and Ruby. This ability to use Selenium-RC with a high level programming language to develop test cases also allows the automated testing to be integrated with a project_s automated build environment.

SELENIUM-GRID

Selenium-Grid allows the Selenium-RC solution to scale for large test suites or test suites that must be run in multiple environments. With Selenium-Grid, multiple instances of Selenium-RC are running on various operating system and browser configurations; Each of these when launching register with a hub. When tests are sent to the hub they are then redirected to an available Selenium-RC, which will launch the browser and run the test. This allows for running tests in parallel, with the entire test suite theoretically taking only as long to run as the longest individual test.

1. Tests developed on Firefox via Selenium-IDE can be executed on any other supported browser via a simple Selenium-RC command line.
2. Selenium-RC server can start any executable, but depending on browser security set-tings there may be technical limitations that would limit certain features.

FLEXIBILITY AND EXTENSIBILITY

Selenium is highly flexible. There are multiple ways in which one can add functionality to Selenium_s framework to customize test automation for one_s specific testing needs. This is, perhaps, Selenium_s strongest characteristic when compared with proprietary test automation tools and other open source solutions. Selenium-RC support for multiple programming and scripting languages allows the test writer to build any logic they need into their automated testing and to use a preferred programming or scripting language of one_s choice.

Selenium-IDE allows for the addition of user-defined user extensions for creating additional commands customized to the user_s needs. Also, it is possible to re-configure how the Selenium-IDE generates its Selenium-RC code. This allows users to customize the generated code to fit in with their own test frameworks. Finally, Selenium is an Open Source project where code can be modified and enhancements can be submitted for contribution.

TEST SUITES

A test suite is a collection of tests. Often one will run all the tests in a test suite as one continuous batch job. When using Selenium -IDE, test suites also can be defined using a simple HTML file. The syntax again is simple. An HTML table defines a list of tests where each row defines the file system path to each test. An example tells it all.

```
<html>
<head>
<title>Test Suite Function Tests – Priority 1</title></head>
<body>
<table>
<tr><td><b>Suite Of Tests</b></td></tr>
<tr><td><a href=./Login.html>Login</a></td></tr>
<tr><td><a href=./SearchValues.html>Test Searching for Values</a></td></tr>
<tr><td><a href=./SaveValues.html>Test Save</a></td></tr>
</table></body>
</html>
```

A file similar to this would allow running the tests all at once, one after another, from the Selenium-IDE.

Test suites can also be maintained when using Selenium-RC. This is done via programming and can be done a number of ways. Commonly Junit is used to maintain a test suite if one is using Selenium-RC with Java. Additionally, if C# is the chosen language, Nunit could be employed. If using an interpreted language like Python with Selenium-RC than some simple programming

would be involved in setting up a test suite. Since the whole reason for using Sel-RC is to make use of programming logic for your testing this usually isn_t a problem.

Few typical Selenium commands.

1. **open –** opens a page using a URL.
2. **click/clickAndWait –** performs a click operation, and optionally waits for a new page toload.
3. **verifyTitle/assertTitle –** verifies an expected page title.
4. **verifyTextPresent –** verifies expected text is somewhere on the page.
5. **verifyElementPresent –** verifies an expected UI element, as defined by its HTML tag, is present on the page.
6. **verifyText –** verifies expected text and it_s corresponding HTML tag are present on thepage.
7. **verifyTable –** verifies a table_s expected contents.
8. **waitForPageToLoad –** pauses execution until an expected new page loads. Called automatically when clickAndWait is used.
9. **waitForElementPresent –** pauses execution until an expected UI element, as defined by its HTML tag, is present on the page.

7.3 VIVA QUESTIONS:

- ✓ Explain about selenium tool?
- ✓ Compare win runner and selenium tool?
- ✓ What are the components of selenium tool?
- ✓ What are test suits for selenium tool?
- ✓ Define selenium grid?

EXPERIMENT # 8

BUGBIT

8.1 OBJECTIVE :

Study of Any Test Management Tool (Test Director , BUGBIT)

8.2 STUDY OF TEST DIRECTOR AND TEST MANAGEMENT TOOL:

Test Director is a global test management solution which provides communication, organization, documentation and structure to the testing project.

Test Director is used for

1. Mapping Requirements to User acceptance test cases
2. Test Planning by placing all the test cases and scripts in it.
3. Manual testing by defining test steps and procedures
4. Test Execution status
5. Defect Management

The Test Director Testing Process

Test Director offers an organized framework for testing applications before they are deployed. Since test plans evolve with new or modified application requirements, you need a central data repository for organizing and managing the testing process. TestDirector guides through the requirements specification, test planning, test execution, and defect tracking phases of the testing process. The Test Director testing process includes four phases:

Specifying Requirements

1. Requirements are linked to tests and defects to provide complete traceability and aid the decision- making process
2. See what percent of requirements are covered by tests
3. Each requirement in the tree is described in detail, and can include any relevant attachments. TheQA tester assigns the requirement a priority level which is taken into consideration when the test team creates the test plan
4. Import from Microsoft Word or third party RM tool

Planning Tests

1. The Test Plan Manager enables to divide application according to functionality. Application can be divided into units, or subjects, by creating a test plan tree.
2. Define subjects according to:
3. Application functionality-such as editing, file operations, and reporting
4. Type of testing-such as functional, user interface, performance, and load

5. As the tests are also linked to defects, this helps ensure compliance with testing requirements throughout the testing process.

Running Tests

As the application constantly changes, using test lab, run manual and automated tests in the project in order to locate defects and assess quality.

1. By creating test sets and choosing which tests to include in each set, test suite can be created? A test set is a group of tests in a Test Director Project database designed to achieve specific testing goals.

Tests can be run manually or scheduled to run automatically based on application dependencies

Tracking Defects

Locating and repairing application defects efficiently is essential to the testing process. Defects can be detected and added during all stages of the testing process. In this phase you per-form the following tasks:

1. This tool features a sophisticated mechanism for tracking software defects, enabling Testing Team and the project Team to monitor defects closely from initial detection until resolution
2. By linking Test Director to e-mail system, defect tracking information can be shared by all Development and Management Teams, Testing and Wipro Software Quality Assurancepersonnel

8.3 VIVA QUESTIONS:

1. Define test director?
2. What are the uses of test director?
3. What is testing process for test director?
4. What are the specification requirements for test director?
5. What are the test plans for test director?

EXPERIMENT # 9

BUG TRACKING TOOL

9.1 OBJECTIVE:

Study of Any Bug Tracking Tool (Bugzilla)

9.2 STUDY OF BUGZILLA,BUGBIT AND BUG TRACKING TOOL:

Bugzilla is a Bug Tracking System that can efficiently keep track of outstanding bugs in a product. Multiple users can access this database and query, add and manage these bugs. Bugzilla essentially comes to the rescue of a group of people working together on a product as it enables them to view current bugs and make contributions to resolve issues. Its basic repository nature works out better than the mailing list concept and an organized database is always easier to work with.

Advantage of Using Bugzilla:

1. Bugzilla is very adaptable to various situations. Known uses currently include IT support queues, Systems Administration deployment management, chip design and development problem tracking (both pre-and-post fabrication), and software and hardware bug tracking for luminaries such as Redhat, NASA, Linux-Mandrake, and VA Systems. Combined with systems such as CVS, Bugzilla provides a powerful, easy to use solution to configuration management and replication problems.

2. Bugzilla can dramatically increase the productivity and accountability of individual employees by providing a documented workflow and positive feedback for good performance. Ultimately, Bugzilla puts the power in user_s hands to improve value to business while providing a usable frameworkfor natural attention to detail and knowledge store to flourish.

The bugzilla utility basically allows to do the following:

1. Add a bug into the database
2. Review existing bug reports
3. Manage the content

Bugzilla is organized in the form of bug reports that give all the information needed about a particular bug. A bug report would consist of the following fields.

1. Product–>Component
2. Assigned to
3. Status (New, Assigned, Fixed etc)
4. Summary

5. Bug priority
6. Bug severity (blocker, trivial etc)
7. Bug reporter

Using Bugzilla:

Bugzilla usage involves the following activities Setting Parameters and Default Preferences

1. Creating a New User
2. Impersonating a User
3. Adding Products
4. Adding Product Components
5. Modifying Default Field Values
6. Creating a New Bug
7. Viewing Bug Reports

Setting Parameters and Default Preferences:
When we start using Bugzilla, we_ll need to set a small number of parameters and preferences. At a minimum, we should change the following items, to suit our particular need:

1. Set the maintainer
2. Set the mail_delivery_method
3. Set bug change policies
4. Set the display order of bug reports

To set parameters and default preferences:

1. Click Parameters at the bottom of the page.
2. Under Required Settings, add an email address in the maintainer field.
3. Click Save Changes.
4. In the left side Index list, click Email.
5. Select from the list of mail transports to match the transport we_re using. If evaluating a click2try application, select test. If using SMTP, set any of the other SMTP options for your environment. Click Save Changes.
6. In the left side Index list, click Bug Change Policies.
7. Select On for comment on create, which will force anyone who enters a new bug to enter a comment, to describe the bug. Click Save Changes.
8. Click Default Preferences at the bottom of the page.
9. Select the display order from the drop-down list next to the When viewing a bug, show comments in this order field. Click Submit Changes.

CREATING A NEW USER

Before entering bugs, make sure we add some new users. We can enter users very easily, with a minimum of information. Bugzilla uses the email address as the user ID, because users are frequently notified when a bug isentered, either becausethey entered the

bug, because the bug is assigned to them, or because they_ve chosen to track bugs in a certain project.

To create a new user:

1. Click **users**.
2. Click **adds** a new user.
3. Enter the **login name**, in the form of an email address.
4. Enter the **real name**, a password, and then click **add**.
5. Select the **group access options**. We_ll probably want to enable the following options in the row titled user is a member of these groups:
6. Can confirm
7. Edit bugs
8. Edit components
9. Click **update** when done with setting options.

Impersonating a User:

Impersonating a user is possible, though rare, that we may need to file or manage a bug in an area that is the responsibility of another user when that user is not available. Perhaps the user is on vacation, or is temporarily assigned to another project. We can impersonate the user to create or manage bugs that belong to that user.

Adding Products

We_ll add a product in Bugzilla for every product we are developing. To start with, when we first login to Bugzilla, we_ll find a test product called **TestProduct**. We should delete this and create a new product.

To add a product:

1. At the bottom of the page, click **Products**.
2. In the **TestProduct** listing, click **Delete**.
3. Click **Yes, Delete**.
4. Now click **Add a product**.
5. Enter a product name, such as Widget Design Kit.
6. Enter a description.

Viewing Bug Reports

Eventually, we_ll end up with thousands of bugs listed in the system. There are several ways to view the bugs. The easiest is to click the My Bugs link at the bottom of the page. Because we_ve only got one bug reported, we_ll use the standard Search function.

To find a bug:

1. Click **Reports**.
2. Click the **Search** link on the page, not the one in the top menu. This opens a page titled Find a Specific Bug.
3. Select the **Status**.
4. Select the **Product**.
5. Enter a word that might be in the title of the bug.
6. Click **Search**. If any bugs meet the criteria that we have entered, Bugzilla displays them in a list summary.
7. Click the **ID** number link to view the full bugreport.

Modifying Bug Reports

Suppose we want to change the status of the bug. We_ve reviewed it and have determined that it belongs to one of the users we have created earlier.

9.3 VIVA QUESTIONS:

- Define bug tracking system?
- Compare hardware and software bug trackingsystem?
- What are the uses of Bugzilla?
- What are the different setting parameters and default preferences?
- Explain how to design test cases using bug bit?

EXPERIMENT # 10

OPEN SOURCE TESTING TOOL

10.1 OBJECTIVE:

Study of any open source testing tool (Test Link).

10.2 STUDY OF TEST LINK OPEN SOURCE TESTING TOOL:

Test link is an open source test management tool. It enables creation and organization of test cases and helps manage into test plan. Allows execution of test cases from test link itself. One can easily track test results dynamically, generate reports, generate test metrics, prioritize test cases and assign unfinished tasks. It‘s a web based tool with GUI, which provides an ease to develop test cases, organize test cases into test plans, execute these test cases and generate re-ports. Test link exposes API, written in PHP, can help generate quality assurance dashboards. The functions like AddTestCase ToTestPlan,

Assign Requirements, Create Test Case etc. helps create and organize test cases per test plan. Functions like GetTestCasesForTestPlan, GetLastExecutionResult allows one to create quality assurance dashboard. TestLink enables easily to create and manage Test cases as well as organize them into Test plans. These Test plans allow team members to execute Test cases and track test results dynamically, generate reports, trace software requirements, prioritize and assign tasks. Read more about implemented features and try demo pages.

Overall structure

There are three cornerstones: **Product**, **Test Plan** and **User**. All other data are relations or attributes for this base. First, definition of a couple of terms that are used throughout the documentation.

Products and Test Plans

1. Product: A Product is something that will exist forever in TestLink. Products will under-go many different versions throughout their lifetimes. Product includes Test Specification with Test Cases and should be sorted via Keywords.

2. Test Plan: Test Plans are created when you_d like to execute test cases. Test plans can be made up of the test cases of one or many Products. Test Plan includes Builds, Test Case Suite and Test Results.

3. User: A User has a Role that defines available Test Link features.

Test Case Categorization

Test Link breaks down the test case structure into three levels Components, Categories, and test cases. These levels are persisted throughout the application.

1. Component: Components are the parents of Categories. Each Component can have many
2. Categories.
3. Category: Categories are the parents of test cases. Each Category can have many test cases.
4. Test Case: Test cases are the fundamental piece of TestLink.
5. Test Specification: All Components, Categories and test cases within Product.
6. Test Case Suite: All Components, Categories and test cases within Test Plan.

Test Specification Creating Test Cases

Tester must follow this structure: Component, Category and test case. At first you create Component(s) for your Product. Component includes Categories. Category has the similar meaning but is second level of Test Specification and includes just Test Cases.User can also copy or move Test Cases.

Test Cases have following parts:

1. Title: could include either short description or abbreviation (e.g. TL-USER-LOGIN)
2. Summary: should be really short; just for overview.
3. Steps: describe test scenario (input actions); can also include precondition and cleanup information here.
4. Expected results: describe checkpoints and expected behavior a tested Product or system.

Deleting Test Cases

Test cases, Categories, and Components may be deleted from a test plan by users with lead permissions from the delete test cases screen. Deleting data may be useful when first creating a test plan since there are no results. However, Deleting test cases will cause the loss of all results associated with them. Therefore, extreme caution is recommended when using this functionality.

Requirements relation

Test cases could be related with software/system requirements as n to n. The functionality must be enabled for a Product. User can assign Test Cases and Requirements via link Assign Requirements in the main screen.

Test Plans

Test plan contains name, description, collection a chosen test cases, builds, test results, milestones, tester assignment and priority definition.

Creating a new Test Plan

Test Plans may be deleted from the Create test plan page (link Create Test Plan) by users with lead privileges. Test plans are the basis for test case execution. Test plans are made up of test cases imported from Products at a specific point of time. Test plans can only be created by users with lead privileges. Test plans may be created from other test plans. This allows users to create test plans from test cases that at a desired point in time. This may be necessary when creating a test plan for a patch. In order for a user to see a test plan they must have the proper rights. Rights may be assigned (by leads) in the define User/Project

Rights section. This is an important thing to remember when users tell you they can_t see the project they are working on.

Test Execution

Test execution is available when:

1. A Test Specification is written.
2. A Test Plan is created.
3. Test Case Suite (for the Test Plan) is defined.
4. A Build is created.
5. The Test plan is assigned to testers (otherwise they cannot navigate to this TestPlan).
6. Select a required Test Plan in main page and navigate to the Execute test link. Left pane serves for navigation in Test Case Suite via tree menu, filtering and define a tested build.

Test Status

Execution is the process of assigning a result (pass, fail, blocked) to a test case for a specific build. Blocked_test case is not possible to test for some reason (e.g. a problem in configuration disallows to run a tested functionality).

Insert Test results

1. Test Results screen is shown via click on an appropriate Component, Category or test case in navigation pane. The title shows the current build and owner. The colored bar indicate status of the test case.
2. Yellow box includes test scenario of the test case.
3. Updated Test Cases: If users have the proper rights they can go to the Update modified testcase page through the link on main page. It is not necessary for users to update test cases if there has been a change (newer version or deleted).

Advantages:

1. Easy in tracking test cases(search with keyword, test case id, version etc)
2. We can add our custom fields to test cases.
3. Allocating the work either test case creation/execution any kind of documents is easy
4. when a test cases is updated the previous version also can be tracked
5. We can generate results build wise
6. Test plans are created for builds and work allocations can be done.

Report, is one of the awesome functionality present in the Test link, it generates reports in desired format like HTML/ CSV /Excel and we can create graphs too. And the above all is done on the privileges based which is an art of the testlink and i liked this featuremuch

Example of TestLink workflow:

1. Administrator create a Product Fast Food and a user Adam with rights leader and Bela with rights Senior tester.
2. Adam imports Software Requirements and for part of these requirements generates empty Test cases.
3. Bela describe test scenario of these Test cases that are organized according to Components and Categories.
4. Adam creates Keyword: Regression and assigns this keyword to ten of these test cases.
5. Adam creates a Test Plan Fish & Chips, Build Fish 0.1 and add Test Cases with keywords Regression.
6. Adam and Bela execute and record the testing with result: 5 passed, 1 failed and 4 are blocked.
7. Developers make a new build Fish 0.2 and Bela tests the failed and blocked test cases only. Exceptionally all these five Test cases passed.
8. Manager would like to see results. Administrator explains him that he can create account himself on the login page. Manager does it. He has Guestrights and could see results and Test cases. He can see that everything passed in overall report and problems in build Fish 0.1 in a report for particular Build. But he can change nothing.

10.3 VIVA QUESTIONS:

1. Define test link?
2. Difference between product and test plan?
3. Explain test case categorization?
4. What is test case specification?
5. How to delete test case in Test Link?