SOFTWARE TESTING METHODOLOGY

LAB MANUAL

Year	**:**	**2019 - 2020**
Course Code	**:**	**AIT104**
Regulations	**:**	**IARE - R16**
Semester	**:**	**VII**
Branch	**:**	**CSE**

Prepared by

GEETAVANI.B, ASSISTANT PROFESSOR

INSTITUTE OF AERONAUTICAL ENGINEERING

(Autonomous)

Dundigal, Hyderabad - 500 043

1. PROGRAM OUTCOMES:

PROGRAM OUTCOMES (POs)	
PO-1:	Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems (**Engineering Knowledge**).
PO-2:	Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences **(Problem Analysis).**
PO-3:	Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations **(Design/Development of Solutions).**
PO-4:	Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions **(Conduct Investigations of Complex Problems).**
PO-5:	Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations **(Modern Tool Usage).**
PO-6:	Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice **(The Engineer and Society).**
PO-7:	Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development **(Environment and Sustainability).**
PO-8:	Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice **(Ethics).**
PO-9:	Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings **(Individual and Team Work).**
PO-10:	Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions **(Communication).**
PO-11:	Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO-12:	Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change **(Life-long learning).**

2. PROGRAM SPECIFIC OUTCOMES (PSOs):

PROGRAM SPECIFIC OUTCOMES (PSO'S)	
PSO – I	**Professional Skills:** The ability to understand, analyze and develop computer programs in the areas related to algorithms, system software, multimedia, web design, big data analytics, and networking for efficient design of computer-based systems of varying complexity.
PSO – II	**Problem-Solving Skills:** The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.
PSO – III	**Successful Career and Entrepreneurship:** The ability to employ modern computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies.

3. ATTAINMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:

Week.No	Experiment	Program Outcomes Attained	Program Specific Outcomes Attained
1.	CONSTRUCTS	PO1, PO2, PO3	PSO2
2.	SYSTEM SPECIFICATIONS	PO1, PO2, PO3	PSO2
3.	TEST CASES	PO1, PO2, PO3	PSO2
4.	TEST PLAN	PO2, PO3	PSO2
5.	TESTING TOOL	PO3, PO4	PSO2
6.	SELENIUM	PO2, PO3	PSO1, PSO2
7.	BUG TRACKING TOOL	PO2, PO3	PSO1, PSO2
8.	BUGBIT	PO2, PO3	PSO1, PSO2
9.	TEST MANAGEMENT TOOL	PO3, PO4	PSO1, PSO2
10.	OPEN SOURCE TESTING TOOL	PO3, PO4	PSO1, PSO2
11.	AUTOMATED FUNCTIONAL TESTING TOOL	PO2, PO3	PSO1, PSO2
12.	INTROSPECTION OF MATRIX MULTIPLICATION	PO2, PO3	PSO1, PSO2

4. MAPPING COURSE OBJECTIVES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES:

Course Objectives (COs)	Program Outcomes (POs)												Program Specific Outcomes (PSOs)		
	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO 1	PSO 2	PSO 3
I	√	√	√										√		
II	√	√		√									√		
III			√										√	√	
IV	√	√	√											√	

5. SYLLABUS:

SOFTWARE TESTING METHODOLOGY LABORATORY

VII Semester: CSE

Course Code	Category	Hours / Week			Credits	Maximum Marks		
		L	T	P	C	CIA	SEE	Total
AIT104	Foundation	-	-	3	2	30	70	100
Contact Classes: Nil	Tutorial Classes: Nil	Practical Classes: 39				Total Classes: 39		

OBJECTIVES:

The course will enable the students to:

I. Learn the importance of web testing tool and bug tracking tool.
II. Develop test case and test plan document for banking application.
III. Learn to write system specifications of any application and report various bugs in it.
IV. Use automated functional testing tool like Quick Test Professional.

LIST OF EXPERIMENTS

WEEK-1 CONSTRUCTS

Write programs in C language to demonstrate the working of the following constructs:

a. while b) switch c) for d) if-else e) do-while

WEEK-2 SYSTEM SPECIFICATIONS

a. Study the system specifications of ATM system and report various bugs in it.

b. Study the system specifications of banking application and report various bugs in it.

WEEK-3 TEST CASES

a. Write the test cases for ATM system.

b. Write the test cases for banking application.

WEEK-4	**TEST PLAN**
Create a test plan document for any application (e.g. Library management system).	
WEEK-5	**TESTING TOOL**
Study of any testing tool (e.g. Win runner).	
WEEK-6	**SELENIUM**
Study of web testing tool (e.g. Selenium).	
WEEK-7	**BUG TRACKING TOOL**
Study of bug tracking tool (e.g. Bugzilla).	
WEEK-8	**BUGBIT**
Study of bug tracking tool (e.g. Bugbit).	
WEEK-9	**TEST MANAGEMENT TOOL**
Study of any test management tool (e.g. Testdirector).	
WEEK-10	**OPEN SOURCE TESTING TOOL**
Study of any Open Source Testing Tool (e.g. Test Link).	
WEEK-11	**AUTOMATED FUNCTIONAL TESTING TOOL**
Study of QTP (Quick Test Professional) automated functional testing tool.	
WEEK-12	**INTROSPECTION OF MATRIX MULTIPLICATION**

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.

Reference Books:

1. Boris Beizer, —Software Testing Techniques‖, DreamTech Press, 2 nd Edition, 2000.
2. Dr. K. V. K. K. Prasad, —Software Testing Tools‖, DreamTech Press, Revised Edition, 2004.
3. Perry, —Effective methods of Software Testing‖, John Wiley, 2 nd Edition, 1999.

Web References:

1. http://www.scoopworld.in
2. http://www.sxecw.edu.in
3. http://www.technofest2u.blogspot.com

SOFTWARE AND HARDWARE REQUIREMENTS FOR A BATCH OF 36 STUDENTS:

HARDWARE:

Intel Desktop Systems: 36 nos

Printers: 02

SOFTWARE:

System Software: Microsoft Windows 7 Academic Get Genuine Legalization License

6. INDEX:

WEEK-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.

1.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

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("%dǁ", &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: %dl ,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("%dl",&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
1234567891222222222222	12345678912222222213	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
1234567891222222222222	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?

1.8 LAB ASSIGNMENT:

1. Demonstrate the working of nested if in C language?
2. Demonstrate the working of simple if in C language?
3. Design test cases for preprocessor commands in C language?
4. Demonstrate the working of go to statement in C language?
5. Design test cases for structures in C language?

1.9 POST LAB VIVA QUESTIONS:

1. Define model for testing?
2. How to design test cases?
3. Give syntax for if …else statement?
4. Give the syntax for nested if in C statement?
5. Compare different testing techniques?

WEEK - 2
SYSTEM SPECIFICATIONS

2.1 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.

2.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

2.3 BUGS IN ATM SYSTEM:

1. Machine is accepting ATM card.
2. Machine is rejecting expired card.
3. Successful entry of PIN number.
4. Unsuccessful operation due to enter wrong PIN number 3 times.
5. Successful selection of language.
6. Successful selection of account type.
7. Unsuccessful operation due to invalid account type.
8. Successful selection of amount to be withdrawn.
9. Successful withdrawal.
10. Expected message due to amount is greater than day limit.
11. Unsuccessful withdraw operation due to lack of money in ATM.
12. Expected message due to amount to withdraw is greater than possible balance.
13. Unsuccessful withdraw operation due to click cancel after insert card.

2.4 PRE LAB VIVA QUESTIONS:

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

2.6 LAB ASSIGNMENT:

1. What can be bugs for online library?
2. Write down the bugs for hospital management system?
3. What can system specifications for online banking?
4. What can system specifications for online shopping?
5. What can system specifications for hotel management system?

2.7 POST LAB VIVA QUESTIONS:

1. What are test design bugs?
2. Explain about nightmare list?
3. What are levels of testing?
4. Compare small versus large programming?
5. Define pesticide paradox?

WEEK- 3

TEST CASES

3.1 OBJECTIVE:

Study the system specifications of banking applications and report the various bugs in it.

3.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

3.3 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 (and thus 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.

3.4 PRE-LAB VIVA QUESTIONS:

1. What is flow graph testing?
2. Difference between flow graph and control graph?
3. Compare data and coding bugs?
4. Differentiate testing and debugging?

5. Explain complexity barrier?

3.6 LAB ASSIGNMENT:

1. Write test case for Library Application?
2. Write test case for online shopping?
3. Write test case for Google web search?
4. Write test case for Android application?
5. Write test case for ATM?

3.7 POST-LAB VIVA QUESTIONS:

1. Write about design test cases?
2. What are integration bugs?
3. Define system bugs?
4. What are design specifications?
5. What is path testing?

WEEK-4
TEST PLAN

4.1 OBJECTIVE:

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

4.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

4.3 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.

4.5 PRE LAB 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?

4.6 LAB ASSIGNMENT

1. Design test plan document for e-library?
2. Design test plan document for credit card processing?
3. Design test plan document for ticket vending machine?
4. Design test plan document for Airport check-in business model?
5. Design test plan document for school management system?

4.7 POST LAB VIVA QUESTIONS:

1. What are domain bugs?
2. What is meant by predicate coverage?
3. Define path sensitization?
4. What C1, C2 Coverage?
5. Define Link marks and Link counters?

WEEK-5
TESTING TOOL

5.1 OBJECTIVE:

Study of Any Testing Tool(Win Runner)

5.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

5.3 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 – ensure process repeatability and resource independence. Eliminates errors of manual testing. Reduces QA cost – Upfront cost of automated testing is easily recovered over the life-time of the product .Improved testing productivity – test suites can be run earlier and more often Proof of adequate testing .For doing Tedious work – test team members can focus on quality areas.

Win Runner Uses:

1. With Win Runner sophisticated automated tests can be created and run on an application. A series of wizards will be provided to the user, and these wizards can create tests in an automated manner.

2. 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. It will show the errors that were found, and it will also give important information about them. The good news about these tests is that they can be reused many times.

3. Win Runner will test the computer program in a way that is very similar to normal user interactions. This is important, because it ensures a high level of accuracy and realism. Even if an

engineer is not physically present, the Recover manager will troubleshoot any problems that may occur, and this will allow the tests to be completed without errors.

4. The Recover Manager is a powerful tool that can assist users with various scenarios. This is important, especially when important data needs to be recovered.

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

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.

The GUI map consists of files maintained separately from your test scripts. If the user interface of your application changes, you have to update only the GUI map, instead of hundreds of tests. This allows you to easily reuse your Context Sensitive test scripts on future versions of your application.

To run a test, you simply play back the test script. Win Runner emulates a user by moving the mouse pointer over your application, selecting objects, and entering keyboard input. Win Runner reads the object descriptions in the GUI map and then searches in the application being tested for objects matching these descriptions. It can locate objects in a window even if their placement has changed.

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

The Win Runner Testing Process

Testing with Win Runner involves six main stages:

1. Create the GUI Map

The first stage is to create the GUI map so Win Runner can recognize the GUI objects in the application being tested. Use the Rapid Test Script wizard to review the user interface of your application and systematically add descriptions of every GUI object to the GUI map.

Alternatively, you can add descriptions of individual objects to the GUI map by clicking objects while recording a test.

2. Create Tests

Next is creation of test scripts by recording, programming, or a combination of both. While recording tests, insert checkpoints where we want to check the response of the application being tested. We can insert checkpoints that check GUI objects, bitmaps, and databases. During this process, Win Runner captures data and saves it as expected results the expected response of the application being tested.

3. Debug Tests

Run tests in Debug mode to make sure they run smoothly. One can set breakpoints, monitor variables, and control how tests are run to identify and isolate defects. Test results are saved in the debug folder, which can be discarded once debugging is finished. When Win Runner runs a test, it checks each script line for basic syntax errors, like incorrect syntax or missing elements in **If**, **While**, **Switch**, and **For** statements. We can use the **Syntax Check** options (**Tools >Syntax Check**) to check for these types of syntax errors before running your test.

4. Run Tests

Tests can be run in Verify mode to test the application. Each time Win Runner encounters a checkpoint in the test script, it compares the current data of the application being tested to the expected data captured earlier. If any mismatches are found, Win Runner captures them as actual results.

5. View Results

Following each test run, Win Runner displays the results in a report. The report details all the major events that occurred during the run, such as checkpoints, error messages, system messages, or user messages. If mismatches are detected at checkpoints during the test run, we can view the expected results and the actual results from the Test Results window. In cases of bitmap mismatches, one can also view a bitmap that displays only the difference between the expected and actual results.

We can view results in the standard Win Runner report view or in the Unified report view. The Win Runner report view displays the test results in a Windows style viewer. The Unified report view displays the results in an HTML style viewer (identical to the style used for Quick Test Professional test results).

6. Report Defects

If a test run fails due to a defect in the application being tested, one can report information about the defect directly from the Test Results window .This information is sent via e-mail to the quality assurance manager, who tracks the defect until it is fixed.

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** check box.

The Main Win Runner Window

The main Win Runner window contains the following key elements:

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

STANDARD TOOLBAR:

The User toolbar displays the tools you frequently use to create test scripts. By default, the User toolbar is hidden. To display the User toolbar, choose Window>User Toolbar. When you create tests, you can minimize the Win Runner window and work exclusively from the toolbar. The User toolbar is customizable. You choose to add or remove buttons using the Settings > Customize User Toolbar menu option. When you reopen Win Runner, the User toolbar appears as it was when you last closed it. The commands on the Standard toolbar and the User toolbar are described in detail in subsequent lessons.

Note that you can also execute many commands using soft keys. Sof keys are keyboard shortcuts for carrying out menu commands. You can configure the softkey combinations for your keyboard using the Softkey Configuration utility in your Win Runner program group. For more information, see the Win Runner at a Glance chapter in your Win Runner User‘s Guide. Now that you are familiar with the main

Win Runner window, take a few minutes to explore these window components before proceeding to the next lesson.

THE TEST WINDOW

You create and run Win Runner tests in the test window. It contains the following key elements:

1. Test window title bar, with the name of the open test
2. Test script, with statements generated by recording and/or programming in TSL, Mercury Interactive_s Test Script Language.
3. Execution arrow, which indicates the line of the test script being executed during a test run, or the line that will next run if you select the Run from arrow option
4. Insertion point, which indicates where you can insert or edit text.

Experiment-1

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 Win Runner, choose Create > Record—Context Sensitive or click the Record button on the toolbar. From this point on, Win Runner records 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 -2

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

Win Runner waits a set time interval for an application to respond to input. The default wait interval is up to 10 seconds. If the application responds slowly during a test run, Win Runner‘s default wait time may not be sufficient, and the test run may unexpectedly fail. If you discover a synchronization problem between the test and your application, you can either:

Increase the default time that Win Runner waits. To do so, you change the value of the Timeout for Checkpoints and CS Statements option in the Run tab of the General Options dialog box(Settings > General Options). This method affects all your tests and slows down many other Context Sensitive operations. Insert a synchronization point into the test script at the exact point where the problem occurs. A synchronization point tells Win Runner to pause the test run in order to wait for a specified response in the application. This is the recommended method for synchronizing a test with your application. In the following exercises you will:

1. Create a test that opens a new order in the Flight Reservation application and inserts the order into the database
2. Change the synchronization settings
3. Identify a synchronization problem
4. Synchronize the test
5. Run the synchronized test

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.

5.4 PRE-LAB 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?

5.5 LAB ASSIGNMENT:

1. Create a script in win runner in context sensitive mode?
2. What are the steps for synchronizing test in win runner?
3. Generate test cases for library application using win runner?
4. Generate test cases for online shopping?
5. Generate test cases for bank application?

5.6 POST-LAB VIVA QUESTIONS:

1. Define context sensitive mode?
2. What is test script?
3. What is data driver wizard?
4. How to create test in win runner?
5. How to debug test in win runner?

WEEK-6
SELENIUM

6.1 OBJECTIVE:

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

6.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

6.3 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 to load.
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 the page.
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.

6.4 PRE-LAB VIVA QUESTIONS:

1. Explain about selenium tool?
2. Compare win runner and selenium tool?
3. What are the components of selenium tool?
4. What are test suits for selenium tool?
5. Define selenium grid?

6.5 LAB ASSIGNMENT:

1. Generate test cases using selenium tool for library application?
2. Generate test cases using selenium tool for online shopping?
3. Generate test cases using selenium tool for bank application?
4. Generate test cases using selenium tool for ATM application?
5. Generate test cases using selenium tool for Google advance search?

6.6 POST-LAB VIVA QUESTIONS:

1. Define selenium IDE?
2. What is an html file?
3. Define the use of html file in selenium tool?
4. What are the features of selenium tool?
5. What is selenium RC code?

WEEK-7
BUG TRACKING TOOL

7.1 OBJECTIVE:

Study of Any Bug Tracking Tool (Bugzilla)

7.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

7.3 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 framework for 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 is entered, either because they 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.

7. Click **Add**. A message appears that you_ll need to add at least one component

Adding Product Components

Products are comprised of components. Software products, in particular, are typically made up of many functional components, which in turn are made up of program elements, like classes and functions. It_s not unusual in a software development team environment for different individuals to be responsible for the bugs that are reported against a given component. Even if there are other programmers working on that component, it_s not uncommon for one person, either a project lead or manager, to be the gatekeeper for bugs. Often, they will review the bugs as they are reported, in order to redirect them to the appropriate developer or even another team, to review the priority and severity supplied by the reporter, and sometimes to reject bugs as duplicates or enhancement requests, for example.

To add a component:

1. Click the link **add at least one component** in the message that appears after creating a new product.
2. Enter the **Component** name.
3. Enter a **Description**.
4. Enter a **default assignee**. Use one of the users we_ve created. Remember to enter the as signee in the form of an email address.
5. Click **Add**.
6. To add more components, click the name of product in the message that reads edit other components of product <**product name**>.

Modifying Default Field Values

Once we begin to enter new bugs, we_ll see a number of drop down lists containing default values. Some of these may work just fine for our product. Others may not. We can modify the values of these fields, adding new values and deleting old ones. Let_s take a look at the OS category.

To modify default field values:

1. At the bottom of the page, in the **Edit** section, click **Field Values**.
2. Click the link, in this case **OS**, for the field we want to edit. The OS field contains a list of operating system names. We are going to add browsers to this list. In reality, we might create a custom field instead, but for the sake of this example, just add them to the OS list.
3. Click **Add a value**. In the **Value** field, enter IE7.Click **Add**.
4. Click **Add a value** again.
5. In the **Value** field, enter Firefox 3.
6. Click **Add**.
7. Where it reads **add other values for the op_sys field**, click **op_sys**.

8. This redisplays the table. We should now see the two new entries at the top of the table. These values will also appear in the OS drop down list when we create a new bug.

Creating a New Bug:

Creating bugs is a big part of what Bugzilla does best. To create a new bug:

1. In the top menu, click **New**.
2. If we_ve defined more than one component, choose the component from the component list.
3. Select a **Severity** and a **Priority**. **Severity** is self explanatory, but **Priority** is generally assumed to be the lower the number, the higher the priority. So, a **P1** is the highest priority bug, a showstopper.
4. Click the **OS** dropdown list to see the options, including the new browser names we entered.
5. Select one of the options.
6. Enter a summary and a description. We can add any other information of choice, but it is not required by the system, although we may determine that our bug reporting policy requires certain information.
7. Click **Commit**. Bugzilla adds our bug report to the database and displays the detail page for that bug.

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 bug report.

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.

To modify a bug report:

1. Scroll down the full bug description and enter a comment in the **Additional Comments** field.

2. Select Reassign bug to and replace the default user ID with one of the other user IDs you created. It must be in the format of an email address.

7.4 PRE-LAB VIVA QUESTIONS:

1. Define bug tracking system?
2. Compare hardware and software bug tracking system?
3. What are the uses of Bugzilla?
4. What are the different setting parameters and default preferences?
5. Explain how to design test cases using bug bit?

7.6 LAB ASSIGNMENT:

1 Generate and apply Bugzilla bug tracking to android application?
2 Generate and apply Bugzilla bug tracking to library application?
3 Generate and apply Bugzilla bug tracking to bank system?
4 Generate and apply bug bit bug tracking to ATM?
5 Generate and apply bug bit bug tracking to Google web search application?

7.7 POST-LAB VIVA QUESTIONS:

1 How to add components in Bugzilla?
2 How to add default field values in Bugzilla?
3 How to add new bug in Bugzilla?
4 How to add view bug reports in Bugzilla?
5 How to add modify bug reports in Bugzilla?

WEEK-8

BUGBIT

8.1 OBJECTIVE:

Study of Any Test Management Tool (Test Director)

8.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

8.3 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. The QA 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 Assurance personnel

8.4 PRE-LAB 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?

8.5 LAB ASSIGNMENT:

1. Generate test cases using test director for web application?
2. Generate test cases using test director for online shopping?
3. Generate test cases using test director for ATM?
4. Generate test cases using test director for library application?

5. Generate test cases using test director for hospital management system?

8.6 POST-LAB VIVA QUESTIONS:

1. Can we link test director to email system?
2. How can we track defects using test director?
3. Define defect management?
4. Explain test planning in test cases and scripts?
5. What is meant by global test management?

WEEK-9

TEST MANAGEMENT TOOL

9.1 OBJECTIVE:

Study of Any Test Management Tool (Test Director)

9.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

.

9.3 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. The QA 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.

2. 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 Assurance personnel

9.4 PRE-LAB 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?

9.5 LAB ASSIGNMENT:

1. Generate test cases using test director for web application?
2. Generate test cases using test director for online shopping?

3. Generate test cases using test director for ATM?
4. Generate test cases using test director for library application?
5. Generate test cases using test director for hospital management system?

9.6 POST-LAB VIVA QUESTIONS:

1. Can we link test director to email system?
2. How can we track defects using test director?
3. Define defect management?
4. Explain test planning in test cases and scripts?
5. What is meant by global test management?

WEEK-10

OPEN SOURCE TESTING TOOL

10.1 OBJECTIVE:

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

10.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

10.3 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 Test Plan).
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 feature much

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.

10.4 PRE-LAB 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?

10.5 LAB ASSIGNMENT:

1. Generate test cases for library application using Test Link?
2. Generate test cases for ATM application using Test Link?
3. Generate test cases for bank application using Test Link?
4. Generate test cases for online shopping using Test Link?
5. Generate test cases for online credit card processing using Test Link?

10.6 PRE-LAB VIVA QUESTIONS:

1. When a test execution is available in Test Link?
2. Explain about test status
3. What are the advantages of Test Link?
4. How to create test cases in Test Link?
5. Difference between Test Link and other test techniques?

WEEK-11

AUTOMATED FUNCTIONAL TESTING TOOL

11.1 OBJECTIVE:

Study of QTP (Quick Test Professional) automated functional testing tool.

11.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

11.3 STUDY OF QTP AUTOMATED FUNCTIONAL TESTING TOOL:

HP QTP is an automated functional Testing tool that helps testers to execute automated tests in order to identify any errors, defects or gaps in contrary to the expected results of the application under test. It was designed by Mercury Interactive and later on acquired by HP. Full form of QTP is QuickTest Professional while UFT means Unified Functional Testing.

Why QTP is the best testing tool?

It is an icon-based tool that automates the regression and Functional Testing of an application

1. Both technical, as well as a non-technical tester, can use Micro Focus QTP
2. It provides both features- Record as well as Playback
3. We can test Desktop as well as the Web-based applications
4. It allows Business Process Testing (BPT)
5. QTP Testing is based on scripting language VB script
6. Micro Focus's UFT uses VBScript to automate applications
7. It supports the largest pool of software development environments like SAP, Oracle etc..
8. QTP tool helps the testers to perform an automated functional testing uninterrupted.

Advantages of QTP Automation:

1. It supports record and playback
2. It uses an active screen to record scripts and helps tester in referring the screen object properties
3. It has excellent object identification process or mechanism
4. It supports different add-ins like Oracle, Java, SAP, NET, Web Forms, People soft, etc..
5. It allows you to enhance the existing tests even without the AUT through an active screen
6. It supports popular automation frameworks- keyword driven testing approach, modular testing approach, data-driven testing approach, etc..
7. It comes with an inbuilt IDE
8. It can be integrated with Test management tools like Quality Center, Test Director, and Winrunner

9. Different types of suites like Smoke, Regression, Sanity can be easily maintained
10. It supports XML
11. Test reporting is possible through QTP for analysis purpose
12. Easy to maintain

11.4 PRE-LAB 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?

11.5 LAB ASSIGNMENT:

1. Generate test cases for library application using Test Link?
2. Generate test cases for ATM application using Test Link?
3. Generate test cases for bank application using Test Link?
4. Generate test cases for online shopping using Test Link?
5. Generate test cases for online credit card processing using Test Link?

11.6 PRE-LAB VIVA QUESTIONS:

1. When a test execution is available in Test Link?
2. Explain about test status
3. What are the advantages of Test Link?
4. How to create test cases in Test Link?
5. Difference between Test Link and other test techniques?

WEEK-12

INTROSPECTION OF MATRIX MULTIPLICATION

12.1 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‖.

12.2 RESOURCES:

1. Server System configuration: 128 MB of RAM , 500 MB of free disk space, Win NT server, Win server, IIS 5.0, MS Access/Oracle 7.x,8.x,9/MS SQL.
2. Server Client System configuration: 64 MB of RAM , 10 MB of free disk space, Win 95/98/NT/2K/XP, IE 5.

12.3 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 and b.
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

12.4 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.

12.5 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();
}
```

12.6 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	

12.7 PRE-LAB VIVA QUESTIONS:

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

12.8 LAB ASSIGNMENT:

1. Write a C Program to print addition of 3-dimensional matrices?
2. Write a C Program to print student details using structures?
3. Write a C Program to print employee details using pointer to structures?
4. Write a C Program to print student details using pointers?
5. Write a C Program to show the working of double pointers?

12.9 POST-LAB VIVA QUESTIONS:

1. When the mouse pressed event is triggered?
2. When the mouse released event is triggered?
3. When the mouse entered event is triggered?
4. What are different types of listeners?
6. What is component?
7. How to interact with the Java system at runtime?
8. What is the super class of all components of Java?
9. What is a container?