

**Bharati Vidyapeeth Deemed University**  
**College of Engineering, Pune**

**Department of Computer Engineering**

# **LAB MANUAL**

**Class: B.Tech SEM V**

**Subject: Software Testing.**

## **Vision of the Institute**

To be World Class University for Social Transformation Through Dynamic Education.

## **Mission of the Institute**

1. To provide quality technical education with advanced equipments, qualified faculty members, infrastructure to meet needs of profession and society.
2. To provide an environment conducive to innovation, creativity, research and entrepreneurial leadership.
3. To practice and promote professional ethics, transparency and accountability for social community, economic and environmental conditions.

## **Vision of the Department**

To pursue and excel in the endeavor for creating globally recognized computer engineers through quality education.

## **Mission of the Department**

- To impart fundamental knowledge and strong skills conforming to a dynamic curriculum.
- To develop professional, entrepreneurial and research competencies encompassing continuing intellectual growth.
- To strive for producing qualified graduates possessing proficient abilities and ethical responsibilities adapting to the demand of working environment.

## **General Instructions (Laboratory rules/ Practical oriented rules)**

## **General Information (Course Oriented)**

### **Software Testing**

Software testing is an activity to check whether the actual results match the expected results and to ensure that the software system is Defect free. It involves execution of a software component or system component to evaluate one or more properties of interest.

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a white box and Black Box Testing.

## **Objectives of the lab**

1. Testing is a process of executing a program with the intent of finding an error.
2. A good test case is one that has a high probability of finding an as yet undiscovered error.
3. A successful test is one that uncovers an as yet undiscovered error.
4. Understand the purpose of testing while producing the software project
5. Understand how to use Selenium IDE.,
6. Understand how to use Bugzilla, Test Director, Test Link

## **Weekly Plan**

Conduction Time: 2 Hrs Per week

<b>Sr. No.</b>	<b>Experiment Name</b>	<b>Planned Week</b>
1	Introduction to software testing Some theory about Software Testing, where and why it is necessary, methods, STLC.	1 <sup>st</sup> Week
2	Study of Test scenario What is test scenario, what is test case, what is test case plan, test case and scenario sample.	2 <sup>nd</sup> Week
3	Understanding types of software testing Automated testing theory, manual testing theory, difference between them, tools used for both.	3 <sup>rd</sup> Week
4	Implementation of manual testing Consider appropriate example and write down test scenario to implement that example in standard format. Initially explain some theory about example and methods you are going to apply.	4 <sup>th</sup> Week
5	Introduction to automated testing tool. (e.g. Selenium)	5 <sup>th</sup> Week
6	Implementation of automated testing Consider example based on functionality like searching URL, engine, search contents and title.	6 <sup>th</sup> Week
7	Study about role of a tester What is role of a tester? Consider appropriate example and apply role of a tester. With respect to role of a tester, write down test cases for considered example.	7 <sup>th</sup> Week
8	Implementation of defect and management tools. Consider appropriate example to implement defect management in software testing.	8 <sup>th</sup> Week
9	Introduction to some software testing tools. (Open source and Licensed)	9 <sup>th</sup> Week
10	Case study of software testing	10 <sup>th</sup> Week

## **Examination Scheme**

Oral Exam is conducted at the end of semester

## **Marking Scheme**

TW/OR: 01 Credit

Term Work/Oral: 50 Marks

## **Laboratory Usage**

Laboratory of software testing course consist of 20 machines with following configuration.

## **Software Requirements**

### **System configuration:**

1GB of RAM ,

Win NT server, Win 2K server, IIS 5.0,

MSAccess/Oracle 7.x,8.x,9/

MS SQL

Software testing tools:

Selenium Tool, Test Director, Bugzilla, Test Link,

## **Practical Pre-requisite**

Knowledge of Software Engineering, Programming Languages.

## **Program Educational Objectives**

The B. Tech Computer Engineering Programme graduates upon completion of this Programme will able to:

1. Exhibit Competence and professional skills pertinent to the working environment.
2. Continue professional Development through available resources and avenues for career growth.
3. Act with ethical and societal awareness as expected from competent practicing professionals.

## **Program Outcomes**

- a) To apply knowledge of computing and mathematics appropriate to the domain.
- b) To logically define, analyse and solve real world problems
- c) To apply design principles in developing hardware/software systems of varying complexity that meet the specified needs
- d) To interpret and analyse data for providing solutions to complex engineering problems
- e) To use and practice engineering and IT tools for professional growth
- f) To understand and respond to legal and ethical issues involving the use of technology for societal benefits
- g) To develop societal relevant projects using available resources
- h) To exhibit professional and ethical responsibilities
- i) To work effectively as an individual and a team member within the professional environment
- j) To prepare and present technical documents using effective communication skills
- k) To demonstrate effective leadership skills throughout the project management life cycle
- l) To understand the significance of lifelong learning for professional development

## **Course Outcomes**

- 1 Recognize and employ various testing levels according to various roles.
- 2 Design manual functional and boundary test cases and conduct white-box testing program code segment.
- 3 Illustrate and evaluate case tools and document it.
- 4 Identify various testing approaches to different types of software systems and applications.
- 5 Distinguish and relate Test Planning, Budgeting with risk analysis.
- 6 Gain substantial knowledge of various Defect Metrics used in Software testing and Agile Testing Environment.

## **CO-PO mapping**

PO →	a	b	c	d	e	f	g	h	i	j	k	l
CO1				2	1	1		3	3	3	3	1
CO2	1	2	2	3	2			3	3	3		
CO3			2	1	3			3	3	3	3	
CO4		2			2			3	3	3		
CO5	1	1	1	3	2			3	3	3	3	2
CO6	2		1	1	2			3	3	3	1	

### **How outcomes are assessed?**

Outcome	Assignment Number	Level	Proficiency evaluated by
Recognize and employ various testing levels according to various roles.	1,3,4,5,6,7,9,10		Introduction
Design manual functional and boundary test cases and conduct white-box testing program code segment.	2,4		Experiment aim
Illustrate and evaluate case tools and document it	5,6,8,9		Conducting experiments and reporting results
Identify various testing approaches to different types of software systems and applications.	4,5,6,7		Setting up experiments
Distinguish and relate Test Planning, Budgeting with risk analysis.	7		Conducting experiments and reporting results
Gain substantial knowledge of various Defect Metrics used in Software testing and Agile Testing Environment.	8		Conducting experiments and reporting results

### **Design Experience Gained**

Students are able to gain knowledge of software testing methodologies and tools which are used by the industry to identify the defects in the product.

### **Learning Experience Gained**

Students get exposure how to use automated testing tools for identifying defects in the projects also able to acquire knowledge of STLC.

# **Bharati Vidyapeeth Deemed University**

## **College of Engineering, Pune**

### **Department of Computer Engineering**

**Academic Year 2017-18**

**B.Tech. Computer (SEM III)**

#### **List of Assignment** **Subject: Software Testing**

1. Introduction to software testing Some theory about Software Testing, where and why it is necessary, methods, STLC
2. Study of Test scenario What is test scenario, what is test case, what is test case plan, test case and scenario sample
3. Understanding types of software testing Automated testing theory, manual testing theory, difference between them, tools used for both
4. Implementation of manual testing Consider appropriate example and write down test scenario to implement that example in standard format. Initially explain some theory about example and methods you are going to apply.
5. Introduction to automated testing tool (e.g. Selenium)
6. Implementation of automated testing Consider example based on functionality like searching URL, engine, search contents and title.
7. Study about role of a tester What is role of a tester? Consider appropriate example and apply role of a tester. With respect to role of a tester, write down test cases for considered example.
8. Implementation of defect and management tools. Consider appropriate example to implement defect management in software testing.
9. Introduction to some software testing tools. (Open source and Licensed)
10. Case study of software testing

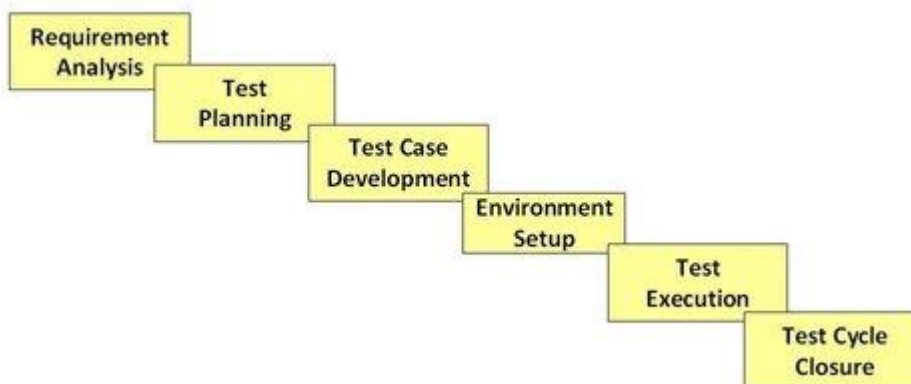
## **Experiment No: 1**

### **Introduction to software testing and STLC.**

**Study assignment which consists of following contents:**

- What is software testing?
- History about software testing, need of testing, manual and automation testing difference, applications of software testing.
- Importance of software testing.
- Why to do testing, quality assurance and quality control.
- Types of software testing.
- Description of various testing types along with example.
- Methods of software testing.
- Software testing methods and strategies to get a quality product.
- Levels of software testing.
- Hierarchical representation of levels of testing
- Software testing life cycle(STLC).

“V” model of software testing along with all phases of STLC.



What is Entry and Exit Criteria?

Entry Criteria: Entry Criteria gives the prerequisite items that must be completed before testing can begin.

Exit Criteria: Exit Criteria defines the items that must be completed before testing can be concluded

You have Entry and Exit Criteria for all levels in the Software Testing Life Cycle (STLC)

In an Ideal world you will not enter the next stage until the exit criteria for the previous stage is met. But practically this is not always possible. So for this tutorial, we will focus on activities and deliverables for the different stages in STLC life cycle. Lets look into them in detail.

### Requirement Analysis

During this phase, test team studies the requirements from a testing point of view to identify the testable requirements.

The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, and System Architects etc) to understand the requirements in detail.

Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance /security availability )

.Automation feasibility for the given testing project is also done in this stage. Activities

- Identify types of tests to be performed.
- Gather details about testing priorities and focus.
- Prepare Requirement Traceability Matrix (RTM).
- Identify test environment details where testing is supposed to be carried out.
- Automation feasibility analysis (if required).

### Deliverables

- RTM
- Automation feasibility report. (if applicable)

### Test Planning

This phase is also called Test Strategy phase. Typically , in this stage, a Senior QA manager will determine effort and cost estimates for the project and would prepare and finalize the Test Plan. Activities

- Preparation of test plan/strategy document for various types of testing
- Test tool selection
- Test effort estimation
- Resource planning and determining roles and responsibilities.

- Training requirement

#### Deliverables

- Test plan /strategy document.
- Effort estimation document.

#### Test Case Development

This phase involves creation, verification and rework of test cases & test scripts. [Test data](#) , is identified/created and is reviewed and then reworked as well.

#### Activities

- Create test cases, automation scripts (if applicable)
- Review and baseline test cases and scripts
- Create test data (If Test Environment is available)

#### Deliverables

- Test cases/scripts
- Test data

#### Test Environment Setup

Test environment decides the software and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and *can be done in parallel with Test Case Development Stage*. *Test team may not be involved in this activity* if the customer/development team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment. Activities

- Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
- Setup test Environment and test data
- Perform smoke test on the build

#### Deliverables

- Environment ready with test data set up
- Smoke Test Results.

#### Test Execution

During this phase test team will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed. Activities

- Execute tests as per plan
- Document test results, and log defects for failed cases
- Map defects to test cases in RTM
- Retest the defect fixes
- Track the defects to closure

#### Deliverables

- Completed RTM with execution status
- Test cases updated with results
- Defect reports

#### Test Cycle Closure

Testing team will meet , discuss and analyze testing artifacts to identify strategies that have to be implemented in future, taking lessons from the current test cycle. The idea is to remove the process bottlenecks for future test cycles and share best practices for any similar projects in future. Activities

- Evaluate cycle completion criteria based on Time,Test coverage,Cost,Software,Critical Business Objectives , Quality
- Prepare test metrics based on the above parameters.
- Document the learning out of the project
- Prepare Test closure report
- Qualitative and quantitative reporting of quality of the work product to the customer.
- Test result analysis to find out the defect distribution by type and severity.

#### Deliverables

- Test Closure report
- Test metrics

#### Summary of STLC Phases along with Entry and Exit Criteria

STLC Stage	Entry Criteria	Activity	Exit Criteria	Deliverables
Requirement Analysis	Requirements Document available (both functional and non functional)	Analyse business functionality to know the business modules and module specific functionalities.	Signed off RTM	RTM
	Acceptance criteria defined.	Identify all transactions in the modules.	Test automation feasibility report signed off by the client	Automation feasibility report (if applicable)
	Application	Identify all the user profiles.		

	architectural document available.	<p>Gather user interface/authentication, geographic spread requirements.</p> <p>Identify types of tests to be performed.</p> <p>Gather details about testing priorities and focus.</p> <p>Prepare Requirement Traceability Matrix (RTM).</p> <p>Identify test environment details where testing is supposed to be carried out.</p> <p>Automation feasibility analysis (if required).</p>		
Test Planning	<p>Requirements Documents</p> <p>Requirement Traceability matrix.</p> <p>Test automation feasibility document.</p>	<p>Analyze various testing approaches available</p> <p>Finalize on the best suited approach</p> <p>Preparation of test plan/strategy document for various types of testing</p> <p>Test tool selection</p> <p>Test effort estimation</p> <p>Resource planning and determining roles and responsibilities.</p>	<p>Approved test plan/strategy document.</p> <p>Effort estimation document signed off.</p>	<p>Test plan/strategy document.</p> <p>Effort estimation document.</p>

Test case development	<p>Requirements Documents</p> <p>RTM and test plan</p> <p>Automation analysis report</p>	<p>Create test cases, automation scripts (where applicable)</p> <p>Review and baseline test cases and scripts</p> <p>Create test data</p>	<p>Reviewed and signed test Cases/scripts</p> <p>Reviewed and signed test data</p>	<p>Test cases/scripts</p> <p>Test data</p>
Test Environment setup	<p>System Design and architecture documents are available</p> <p>Environment set-up plan is available</p>	<p>Understand the required architecture, environment set-up</p> <p>Prepare hardware and software requirement list</p> <p>Finalize connectivity requirements</p> <p>Prepare environment setup checklist</p> <p>Setup test Environment and test data</p> <p>Perform smoke test on the build</p> <p>Accept/reject the build depending on smoke test result</p>	<p>Environment setup is working as per the plan and checklist</p> <p>Test data setup is complete</p> <p>Smoke test is successful</p>	<p>Environment ready with test data set up</p> <p>Smoke Test Results.</p>
Test Execution	<p>Baselined RTM, Test Plan , Test case/scripts are available</p> <p>Test environment is ready</p> <p>Test data set up is</p>	<p>Execute tests as per plan</p> <p>Document test results, and log defects for failed cases</p> <p>Update test plans/test cases, if necessary</p> <p>Map defects to test cases in</p>	<p>All tests planned are executed</p> <p>Defects logged and tracked to closure</p>	<p>Completed RTM with execution status</p> <p>Test cases updated with results</p>

	done	RTM		Defect reports
	Unit/Integration test report for the build to be tested is available	Retest the defect fixes  Regression testing of application  Track the defects to closure		
Test Cycle closure	Testing has been completed  Test results are available  Defect logs are available	Evaluate cycle completion criteria based on - Time, Test coverage , Cost , Software Quality , Critical Business Objectives  Prepare test metrics based on the above parameters.  Document the learning out of the project  Prepare Test closure report  Qualitative and quantitative reporting of quality of the work product to the customer.  Test result analysis to find out the defect distribution by type and severity		

## **Experiment No: 2**

### **Study of test scenario.**

#### **What is Scenario Testing?**

Scenario testing is a software testing technique that makes best use of scenarios. Scenarios help a complex system to test better where in the scenarios are to be credible which are easy to evaluate.

#### **Methods in Scenario Testing:**

- System scenarios
- Use-case and role-based scenarios

#### **Strategies to Create Good Scenarios:**

- Enumerate possible users their actions and objectives
- Evaluate users with hacker's mindset and list possible scenarios of system abuse.
- List the system events and how does the system handle such requests.
- List benefits and create end-to-end tasks to check them.
- Read about similar systems and their behaviour.
- Studying complaints about competitor's products and their predecessor.

#### **Scenario Testing Risks:**

- When the product is unstable, scenario testing becomes complicated.
- Scenario testing are not designed for test coverage.
- Scenario tests are often heavily documented and used time and again

Example:

### **Test Scenarios for Image Upload Functionality**

(Also applicable for other file upload functionality)

1. Check for uploaded image path.
2. Check image upload and change functionality.
3. Check image upload functionality with image files of different extensions (e.g. JPEG, PNG, BMP etc.)
4. Check image upload functionality with images having space or any other allowed special character in the file name.
5. Check duplicate name image upload.
6. Check image upload with image size greater than the max allowed size. The Proper error message should be displayed.
7. Check image upload functionality with file types other than images (e.g. txt, doc, pdf, exe etc.). A Proper error message should be displayed.
8. Check if images of specified height and width (if defined) are accepted otherwise rejected.
9. The image upload progress bar should appear for large size images.
10. Check if cancel button functionality is working in between upload process.
11. Check if file selection dialog shows only supported files listed.
12. Check multiple images upload functionality.
13. Check image quality after upload. Image quality should not be changed after upload.
14. Check if the user is able to use/view the uploaded images.

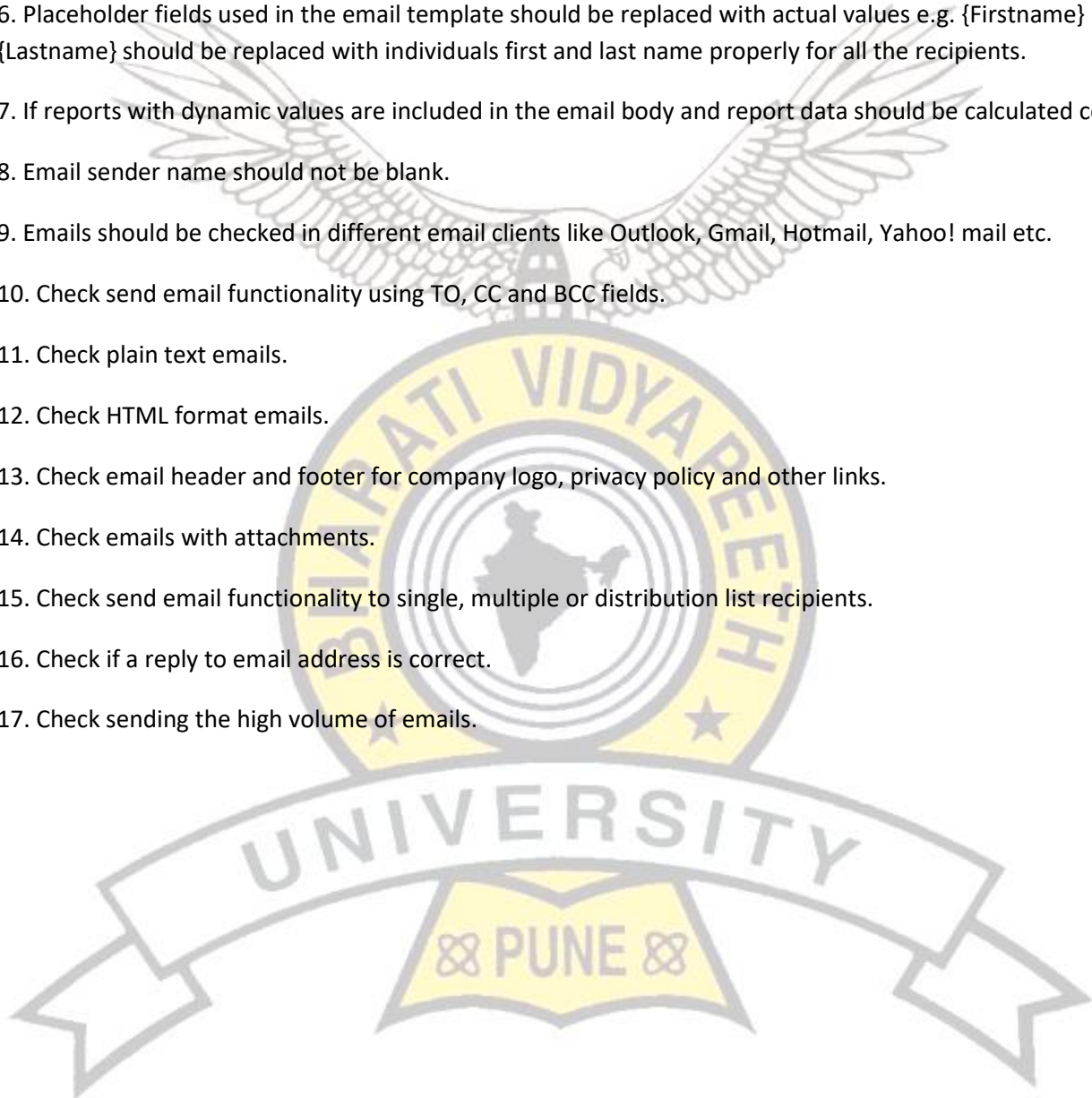
Test Scenarios for Sending Emails

**(Test cases for composing or validating emails are not included here)**

(Make sure to use dummy email addresses before executing email related tests)

1. Email template should use standard CSS for all emails.
2. Email addresses should be validated before sending emails.

3. Special characters in the email body template should be handled properly.
4. Language specific characters (e.g. Russian, Chinese or German language characters) should be handled properly in the email body template.
5. Email subject should not be blank.
6. Placeholder fields used in the email template should be replaced with actual values e.g. {Firstname} {Lastname} should be replaced with individuals first and last name properly for all the recipients.
7. If reports with dynamic values are included in the email body and report data should be calculated correctly.
8. Email sender name should not be blank.
9. Emails should be checked in different email clients like Outlook, Gmail, Hotmail, Yahoo! mail etc.
10. Check send email functionality using TO, CC and BCC fields.
11. Check plain text emails.
12. Check HTML format emails.
13. Check email header and footer for company logo, privacy policy and other links.
14. Check emails with attachments.
15. Check send email functionality to single, multiple or distribution list recipients.
16. Check if a reply to email address is correct.
17. Check sending the high volume of emails.



## **Experiment No:3**

### **Understanding of Types of Software Testing**

#### **Software Testing Standards:**

Many organizations around the globe develop and implement different standards to improve the quality needs of their software. This chapter briefly describes some of the widely used standards related to Quality Assurance and Testing.

#### **ISO/IEC 9126**

This standard deals with the following aspects to determine the quality of a software application:

- Quality model
- External metrics
- Internal metrics
- Quality in use metrics

This standard presents some set of quality attributes for any software such as:

- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

The above-mentioned quality attributes are further divided into sub-factors, which you can study when you study the standard in detail.

## ISO/IEC 9241-11

Part 11 of this standard deals with the extent to which a product can be used by specified users to achieve specified goals with Effectiveness, Efficiency and Satisfaction in a specified context of use.

This standard proposed a framework that describes the usability components and the relationship between them. In this standard, the usability is considered in terms of user performance and satisfaction. According to ISO 9241-11, usability depends on context of use and the level of usability will change as the context changes.

## ISO/IEC 25000:2005

ISO/IEC 25000:2005 is commonly known as the standard that provides the guidelines for Software Quality Requirements and Evaluation (SQuaRE). This standard helps in organizing and enhancing the process related to software quality requirements and their evaluations. In reality, ISO-25000 replaces the two old ISO standards, i.e. ISO-9126 and ISO-14598.

**SQuaRE** is divided into sub-parts such as:

- ISO 2500n - Quality Management Division
- ISO 2501n - Quality Model Division
- ISO 2502n - Quality Measurement Division
- ISO 2503n - Quality Requirements Division
- ISO 2504n - Quality Evaluation Division

The main contents of SQuaRE are:

- Terms and definitions
- Reference Models
- General guide
- Individual division guides
- Standard related to Requirement Engineering (i.e. specification, planning, measurement and evaluation process)

## ISO/IEC 12119

This standard deals with software packages delivered to the client. It does not focus or deal with the clients' production process. The main contents are related to the following items:

- Set of requirements for software packages.
- Instructions for testing a delivered software package against the specified requirements.

### Miscellaneous

Some of the other standards related to QA and Testing processes are mentioned below:

Standard	Description
IEEE 829	A standard for the format of documents used in different stages of software testing.
IEEE 1061	A methodology for establishing quality requirements, identifying, implementing, analyzing, and validating the process, and product of software quality metrics.
IEEE 1059	Guide for Software Verification and Validation Plans.
IEEE 1008	A standard for unit testing.
IEEE 1012	A standard for Software Verification and Validation.
IEEE 1028	A standard for software inspections.
IEEE 1044	A standard for the classification of software anomalies.

IEEE 1044-1	A guide for the classification of software anomalies.
IEEE 830	A guide for developing system requirements specifications.
IEEE 730	A standard for software quality assurance plans.
IEEE 1061	A standard for software quality metrics and methodology.
IEEE 12207	A standard for software life cycle processes and life cycle data.
BS 7925-1	A vocabulary of terms used in software testing.
BS 7925-2	A standard for software component testing.

### Software Testing Types:

#### Manual Testing

Manual testing includes testing a software manually, i.e., without using any automated tool or any script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing.

Testers use test plans, test cases, or test scenarios to test a software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.

#### Automation Testing

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses

another software to test the product. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly, and repeatedly.



Apart from regression testing, automation testing is also used to test the application from load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money in comparison to manual testing.

What is Automate?

It is not possible to automate everything in a software. The areas at which a user can make transactions such as the login form or registration forms, any area where large number of users can access the software simultaneously should be automated.

Furthermore, all GUI items, connections with databases, field validations, etc. can be efficiently tested by automating the manual process.

When to Automate?

Test Automation should be used by considering the following aspects of a software:

- Large and critical projects
- Projects that require testing the same areas frequently
- Requirements not changing frequently
- Accessing the application for load and performance with many virtual users

- Stable software with respect to manual testing
- Availability of time

How to Automate?

Automation is done by using a supportive computer language like VB scripting and an automated software application. There are many tools available that can be used to write automation scripts. Before mentioning the tools, let us identify the process that can be used to automate the testing process:

- Identifying areas within a software for automation
- Selection of appropriate tool for test automation
- Writing test scripts
- Development of test suits
- Execution of scripts
- Create result reports
- Identify any potential bug or performance issues

Software Testing Tools

The following tools can be used for automation testing:

- HP Quick Test Professional
- Selenium
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- Testing Anywhere
- WinRunner
- LoadRunner
- Visual Studio Test Professional
- WATIR

## Software Testing Methods:

There are different methods that can be used for software testing. This chapter briefly describes the methods available.

### Black-Box Testing

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

The following table lists the advantages and disadvantages of black-box testing.

Advantages	Disadvantages
<ul style="list-style-type: none"><li>• Well suited and efficient for large code segments.</li><li>• Code access is not required.</li><li>• Clearly separates user's perspective from the developer's perspective through visibly defined roles.</li><li>• Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.</li></ul>	<ul style="list-style-type: none"><li>• Limited coverage, since only a selected number of test scenarios is actually performed.</li><li>• Inefficient testing, due to the fact that the tester only has limited knowledge about an application.</li><li>• Blind coverage, since the tester cannot target specific code segments or error-prone areas.</li></ul>

- The test cases are difficult to design.

### White-Box Testing

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open-box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

The following table lists the advantages and disadvantages of white-box testing.

Advantages	Disadvantages
<ul style="list-style-type: none"> <li>• As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.</li> <li>• It helps in optimizing the code.</li> <li>• Extra lines of code can be removed which can bring in hidden defects.</li> </ul>	<ul style="list-style-type: none"> <li>• Due to the fact that a skilled tester is needed to perform white-box testing, the costs are increased.</li> <li>• Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems, as many paths will go untested.</li> <li>• It is difficult to maintain white-box testing, as it requires specialized tools like code</li> </ul>

<ul style="list-style-type: none"> <li>• Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.</li> </ul>	analyzers and debugging tools.
--	--------------------------------

### Grey-Box Testing

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

Advantages	Disadvantages
<ul style="list-style-type: none"> <li>• Offers combined benefits of black-box and white-box testing wherever possible.</li> <li>• Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications.</li> <li>• Based on the limited information available, a grey-box tester can design</li> </ul>	<ul style="list-style-type: none"> <li>• Since the access to source code is not available, the ability to go over the code and test coverage is limited.</li> <li>• The tests can be redundant if the software designer has already run a test case.</li> <li>• Testing every possible input stream is unrealistic because it would take an</li> </ul>

<p>excellent test scenarios especially around communication protocols and data type handling.</p> <ul style="list-style-type: none"> <li>The test is done from the point of view of the user and not the designer.</li> </ul>	<p>unreasonable amount of time; therefore, many program paths will go untested.</p>
---	---

### A Comparison of Testing Methods

The following table lists the points that differentiate black-box testing, grey-box testing, and white-box testing.

Black-Box Testing	Grey-Box Testing	White-Box Testing
The internal workings of an application need not be known.	The tester has limited knowledge of the internal workings of the application.	Tester has full knowledge of the internal workings of the application.
Also known as closed-box testing, data-driven testing, or functional testing.	Also known as translucent testing, as the tester has limited knowledge of the insides of the application.	Also known as clear-box testing, structural testing, or code-based testing.
Performed by end-	Performed by end-users	Normally done by

users and also by testers and developers.	and also by testers and developers.	testers and developers.
Testing is based on external expectations - Internal behavior of the application is unknown.	Testing is done on the basis of high-level database diagrams and data flow diagrams.	Internal workings are fully known and the tester can design test data accordingly.
It is exhaustive and the least time-consuming.	Partly time-consuming and exhaustive.	The most exhaustive and time-consuming type of testing.
Not suited for algorithm testing.	Not suited for algorithm testing.	Suited for algorithm testing.
This can only be done by trial-and-error method.	Data domains and internal boundaries can be tested, if known.	Data domains and internal boundaries can be better tested.

### Levels of Software Testing:

There are different levels during the process of testing. In this chapter, a brief description is provided about these levels.

Levels of testing include different methodologies that can be used while conducting software testing.

The main levels of software testing are:

- Functional Testing
- Non-functional Testing

#### Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of a software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

There are five steps that are involved while testing an application for functionality.

Steps	Description
I	The determination of the functionality that the intended application is meant to perform.
II	The creation of test data based on the specifications of the application.
III	The output based on the test data and the specifications of the application.
IV	The writing of test scenarios and the execution of test cases.
V	The comparison of actual and expected results based on the executed test cases.

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality.

### Unit Testing

This type of testing is performed by developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is different from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

### Limitations of Unit Testing

Testing cannot catch each and every bug in an application. It is impossible to evaluate every execution path in every software application. The same is the case with unit testing.

There is a limit to the number of scenarios and test data that a developer can use to verify a source code. After having exhausted all the options, there is no choice but to stop unit testing and merge the code segment with other units.

### Integration Testing

Integration testing is defined as the testing of combined parts of an application to determine if they function correctly. Integration testing can be done in two ways: Bottom-up integration testing and Top-down integration testing.

S.N.	Integration Testing Method
1	Bottom-up integration  This testing begins with unit testing, followed by tests of

	progressively higher-level combinations of units called modules or builds.
2	<p>Top-down integration</p> <p>In this testing, the highest-level modules are tested first and progressively, lower-level modules are tested thereafter.</p>

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic actual situations.

### System Testing

System testing tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets the specified Quality Standards. This type of testing is performed by a specialized testing team.

System testing is important because of the following reasons:

- System testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment that is very close to the production environment where the application will be deployed.
- System testing enables us to test, verify, and validate both the business requirements as well as the application architecture.

### Regression Testing

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.

Regression testing is important because of the following reasons:

- Minimize the gaps in testing when an application with changes made has to be tested.
- Testing the new changes to verify that the changes made did not affect any other area of the application.
- Mitigates risks when regression testing is performed on the application.
- Test coverage is increased without compromising timelines.
- Increase speed to market the product.

#### Acceptance Testing

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors, or interface gaps, but also to point out any bugs in the application that will result in system crashes or major errors in the application.

By performing acceptance tests on an application, the testing team will deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

#### Alpha Testing

This test is the first stage of testing and will be performed amongst the teams (developer and QA

teams). Unit testing, integration testing and system testing when combined together is known as alpha testing. During this phase, the following aspects will be tested in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

#### Beta Testing

This test is performed after alpha testing has been successfully performed. In beta testing, a sample of the intended audience tests the application. Beta testing is also known as **pre-release testing**. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase, the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release it to the general public will increase customer satisfaction.

#### Non-Functional Testing

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing a software from the requirements which are nonfunctional in nature but important such as performance, security, user interface, etc.

Some of the important and commonly used non-functional testing types are discussed below.

### Performance Testing

It is mostly used to identify any bottlenecks or performance issues rather than finding bugs in a software. There are different causes that contribute in lowering the performance of a software:

- Network delay
- Client-side processing
- Database transaction processing
- Load balancing between servers
- Data rendering

Performance testing is considered as one of the important and mandatory testing type in terms of the following aspects:

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

Performance testing can be either qualitative or quantitative and can be divided into different sub-types such as **Load testing** and **Stress testing**.

### Load Testing

It is a process of testing the behavior of a software by applying maximum load in terms of software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of software and its behavior at peak time.

Most of the time, load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test, etc.

Virtual users (VUsers) are defined in the automated testing tool and the script is executed to verify the load testing for the software. The number of users can be increased or decreased concurrently or incrementally based upon the requirements.

### Stress Testing

Stress testing includes testing the behavior of a software under abnormal conditions. For example, it may include taking away some resources or applying a load beyond the actual load limit.

The aim of stress testing is to test the software by applying the load to the system and taking over the resources used by the software to identify the breaking point. This testing can be performed by testing different scenarios such as:

- Shutdown or restart of network ports randomly
- Turning the database on or off
- Running different processes that consume resources such as CPU, memory, server, etc.

### Usability Testing

Usability testing is a black-box technique and is used to identify any error(s) and improvements in the software by observing the users through their usage and operation.

According to Nielsen, usability can be defined in terms of five factors, i.e. efficiency of use, learnability, memory-ability, errors/safety, and satisfaction. According to him, the usability of a product will be good and the system is usable if it possesses the above factors.

Nigel Bevan and Macleod considered that usability is the quality requirement that can be measured as the outcome of interactions with a computer system. This requirement can be fulfilled and the end-user will be satisfied if the intended goals are achieved effectively with the use of proper resources.

Molich in 2000 stated that a user-friendly system should fulfill the following five goals, i.e., easy to

Learn, easy to remember, efficient to use, satisfactory to use, and easy to understand.

In addition to the different definitions of usability, there are some standards and quality models and methods that define usability in the form of attributes and sub-attributes such as ISO-9126, ISO-9241-11, ISO-13407, and IEEE std.610.12, etc.

### UI vs Usability Testing

UI testing involves testing the Graphical User Interface of the Software. UI testing ensures that the GUI functions according to the requirements and tested in terms of color, alignment, size, and other properties.

On the other hand, usability testing ensures a good and user-friendly GUI that can be easily handled. UI testing can be considered as a sub-part of usability testing.

### Security Testing

Security testing involves testing a software in order to identify any flaws and gaps from security and vulnerability point of view. Listed below are the main aspects that security testing should ensure:

- Confidentiality
- Integrity
- Authentication
- Availability
- Authorization
- Non-repudiation
- Software is secure against known and unknown vulnerabilities
- Software data is secure
- Software is according to all security regulations
- Input checking and validation
- SQL insertion attacks
- Injection flaws

- Session management issues
- Cross-site scripting attacks
- Buffer overflows vulnerabilities
- Directory traversal attacks

#### Portability Testing

Portability testing includes testing a software with the aim to ensure its reusability and that it can be moved from another software as well. Following are the strategies that can be used for portability testing:

- Transferring an installed software from one computer to another.
- Building executable (.exe) to run the software on different platforms.

Portability testing can be considered as one of the sub-parts of system testing, as this testing type includes overall testing of a software with respect to its usage over different environments. Computer hardware, operating systems, and browsers are the major focus of portability testing. Some of the pre-conditions for portability testing are as follows:

- Software should be designed and coded, keeping in mind the portability requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.



## **Experiment No: 4**

### **Introduction to Automated Testing Tools**

#### **What is an Automated Software Testing?**

Software Test automation makes use of specialized tools to control the execution of tests and compares the actual results against the expected result. Usually regression tests, which are repetitive actions, are automated.

Testing Tools not only help us to perform regression tests but also helps us to automate data set up generation, product installation, GUI interaction, defect logging, etc.

Criteria for Tool Selection:

For automating any application, the following parameters should be considered.

- Data driven capabilities
- Debugging and logging capabilities
- Platform independence
- Extensibility & Customizability
- E-mail Notifications
- Version control friendly
- Support unattended test runs

**Types of Frameworks:**

Typically, there are 4 test automation frameworks that are adopted while automating the applications.

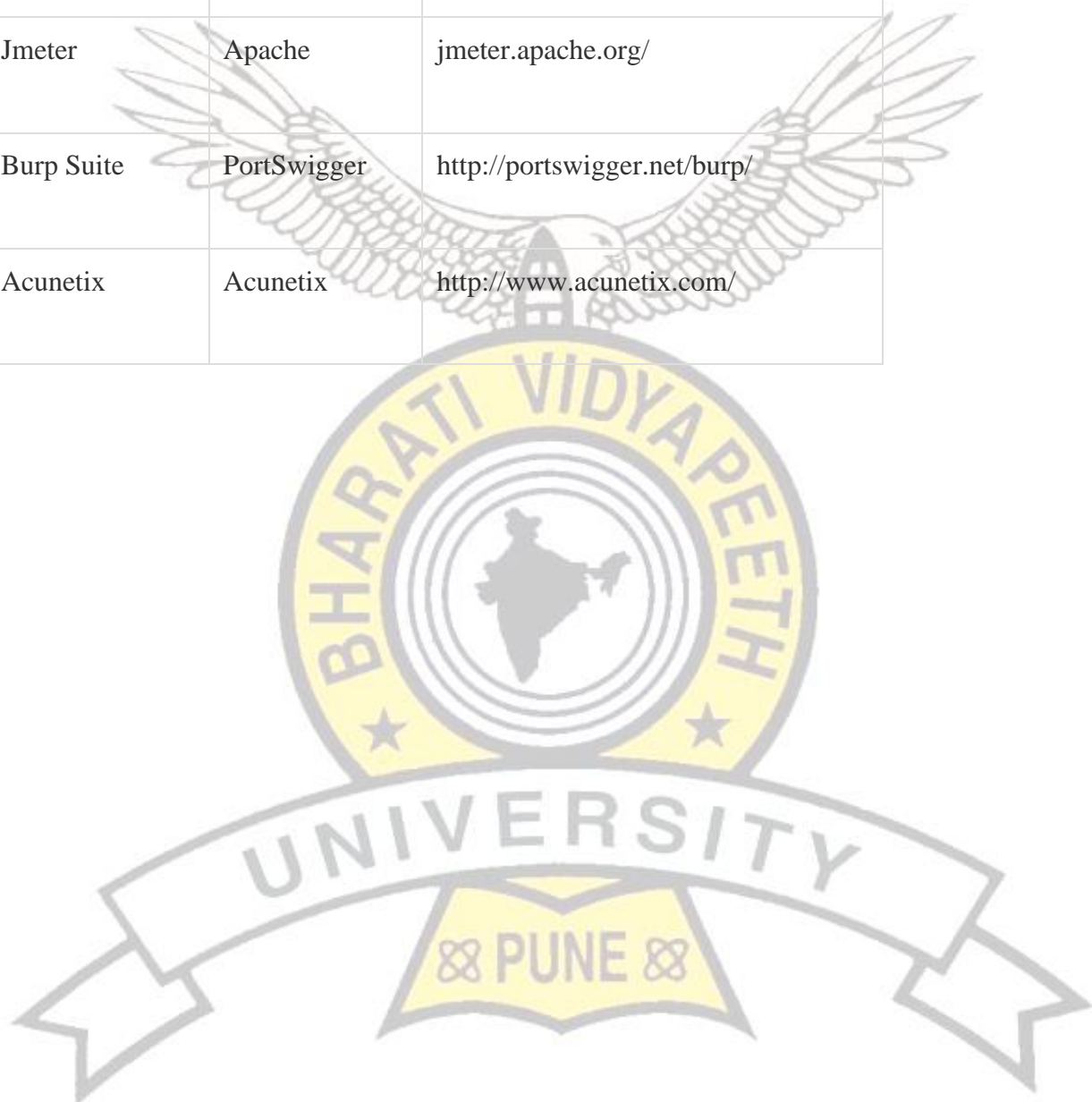
- Data Driven Automation Framework
- Keyword Driven Automation Framework
- Modular Automation Framework
- Hybrid Automation Framework

**Popular Tools that are used for Functional automation:**

Product	Vendor	URL
Quick Test Professional	HP	<a href="http://www.hp.com/go/qtp">www.hp.com/go/qtp</a>
Rational Robot	IBM	<a href="http://www-03.ibm.com/software/products/us/en/robot/">http://www-03.ibm.com/software/products/us/en/robot/</a>
Coded UI	Microsoft	<a href="http://msdn.microsoft.com/en-us/library/dd286726.aspx">http://msdn.microsoft.com/en-us/library/dd286726.aspx</a>
Selenium	Open Source	<a href="http://docs.seleniumhq.org/">http://docs.seleniumhq.org/</a>
Auto IT	Open Source	<a href="http://www.autoitscript.com/site/">http://www.autoitscript.com/site/</a>

**Popular Tools that are used for Non-Functional automation:**

Product	Vendor	URL
Load Runner	HP	<a href="http://www.hp.com/go/LoadRunner">www.hp.com/go/LoadRunner</a>
Jmeter	Apache	<a href="http://jmeter.apache.org/">jmeter.apache.org/</a>
Burp Suite	PortSwigger	<a href="http://portswigger.net/burp/">http://portswigger.net/burp/</a>
Acunetix	Acunetix	<a href="http://www.acunetix.com/">http://www.acunetix.com/</a>



## **Experiment No:5**

### **Implementation of Testing Tool: Selenium**

Selenium is an open-source and a portable automated software testing tool for testing web applications. It has capabilities to operate across different browsers and operating systems. Selenium is not just a single tool but a set of tools that helps testers to automate web-based applications more efficiently.

Let us now understand each one of the tools available in the Selenium suite and their usage.

<b>Tool</b>	<b>Description</b>
Selenium IDE	Selenium Integrated Development Environment (IDE) is a Firefox plugin that lets testers to record their actions as they follow the workflow that they need to test.
Selenium RC	Selenium Remote Control (RC) was the flagship testing framework that allowed more than simple browser actions and linear execution. It makes use of the full power of programming languages such as Java, C#, PHP, Python, Ruby and PERL to create more complex tests.
Selenium WebDriver	Selenium WebDriver is the successor to Selenium RC which sends commands directly to the browser

	and retrieves results.
Selenium Grid	Selenium Grid is a tool used to run parallel tests across different machines and different browsers simultaneously which results in minimized execution time.

### Advantages of Selenium

QTP and Selenium are the most used tools in the market for software automation testing. Hence it makes sense to compare the pros of Selenium over QTP.

Selenium	QTP
Selenium is an open-source tool.	QTP is a commercial tool and there is a cost involved in each one of the licenses.
Can be extended for various technologies that expose DOM.	Limited add-ons and needs add-ons for each one of the technologies.
Has capabilities to execute scripts across different browsers.	Can run tests in specific versions of Firefox , IE, and Chrome.
Can execute scripts on various operating systems.	Works only with Windows.

Supports mobile devices.	Supports mobile devices with the help of third-party tools.
Executes tests within the browser, so focus is NOT required while script execution is in progress.	Needs Focus during script execution, as the tool acts on the browser (mimics user actions).
Can execute tests in parallel with the use of Selenium Grids.	QTP cannot execute tests in parallel, however integrating QTP with QC allows testers to execute in parallel. QC is also a commercial tool.

### Disadvantages of Selenium

Let us now discuss the pitfalls of Selenium over QTP.

Selenium	QTP
Supports only web based applications.	Can test both web and desktop applications.
No feature such as Object Repository/Recovery Scenario	QTP has built-in object repositories and recovery scenarios.
No IDE, so the script development won't be as fast as QTP.	More intuitive IDE; automation can be achieved faster.

Cannot access controls within the browser.	Can access controls within the browser such as favorites bar, backward, and forward buttons.
No default test report generation.	Default test result generation within the tool.
For parameterization, users has to rely on the programming language.	Parameterization is built-in and easy to implement.

The Selenium-IDE (Integrated Development Environment) is an easy-to-use Firefox plug-in to develop Selenium test cases. It provides a Graphical User Interface for recording user actions using Firefox which is used to learn and use Selenium, but it can only be used with Firefox browser as other browsers are not supported.

However, the recorded scripts can be converted into various programming languages supported by Selenium and the scripts can be executed on other browsers as well.

The following table lists the sections that we are going to cover in this chapter.

Title	Description
<b>Download Selenium IDE</b>	This section deals with how to download and configure Selenium IDE.
<b>Selenium IDE Features</b>	This section deals with the features available in Selenium IDE.

<b>Creating Selenium IDE Tests</b>	This section deals with how to create IDE tests using recording feature.
<b>Selenium IDE Script Debugging</b>	This section deals with debugging the Selenium IDE script.
<b>Inserting Verification Points</b>	This section describes how to insert verification points in Selenium IDE.
<b>Selenium Pattern Matching</b>	This section deals with how to work with regular expressions using IDE.
<b>Selenium User Extensions</b>	The Java script that allows users to customize or add new functionality.
<b>Different Browser Execution</b>	This section deals with how to execute Selenium IDE scripts on different browsers.

### **Selenium - Environment Setup**

In order to develop Selenium RC or WebDriver scripts, users have to ensure that they have the initial configuration done. Setting up the environment involves the following steps.

- **Download and Install Java**
- **Download and Configure Eclipse**
- **Configure FireBug and FirePath**
- **Configure Selenium RC**
- **Configure Selenium WebDriver**

#### **Download and Install Java**

We need to have JDK (Java Development Kit) installed in order to work with Selenium

WebDriver/Selenium. Let us see how to download and install Java.

**Step 1 :** Navigate to the URL: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>

**Step 2 :** Go to "Downloads" section and select "JDK Download".



**Step 3 :** Select "Accept License Agreement" radio button.

## Java SE Development Kit 8 Downloads

Thank you for downloading this release of the Java™ Platform, Standard Edition Development Kit (JDK™). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

See also:

- Java Developer Newsletter (tick the checkbox under Subscription Center > Oracle Technology News)
- Java Developer Day hands-on workshops (free) and other events
- Java Magazine

JDK MD5 Checksum












**Looking for JDK 8 on ARM?**

JDK 8 for ARM downloads have moved to the JDK 8 for ARM download page.

### Java SE Development Kit 8u11

You must accept the [Oracle Binary Code License Agreement for Java SE](#) to download this software.

☒ Accept License Agreement ☐ Decline License Agreement












Product / File Description	File Size	Download
Linux x86	133.58 MB	 <a href="#">jdk-8u11-linux-i586.rpm</a>
Linux x86	152.55 MB	 <a href="#">jdk-8u11-linux-i586.tar.gz</a>
Linux x64	133.89 MB	 <a href="#">jdk-8u11-linux-x64.rpm</a>
Linux x64	151.65 MB	 <a href="#">jdk-8u11-linux-x64.tar.gz</a>
Mac OS X x64	207.82 MB	 <a href="#">jdk-8u11-macosx-x64.dmg</a>
Solaris SPARC 64-bit (SVR4 package)	135.66 MB	 <a href="#">jdk-8u11-solaris-sparcv9.tar.Z</a>
Solaris SPARC 64-bit	96.14 MB	 <a href="#">jdk-8u11-solaris-sparcv9.tar.gz</a>
Solaris x64 (SVR4 package)	135.7 MB	 <a href="#">jdk-8u11-solaris-x64.tar.Z</a>
Solaris x64	93.18 MB	 <a href="#">jdk-8u11-solaris-x64.tar.gz</a>
Windows x86	151.81 MB	 <a href="#">jdk-8u11-windows-i586.exe</a>
Windows x64	155.29 MB	 <a href="#">jdk-8u11-windows-x64.exe</a>

**Step 4 :** Select the appropriate installation. In this case, it is 'Windows 7-64' bit. Click the appropriate link and save the .exe file to your disk.

### Java SE Development Kit 8u11

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

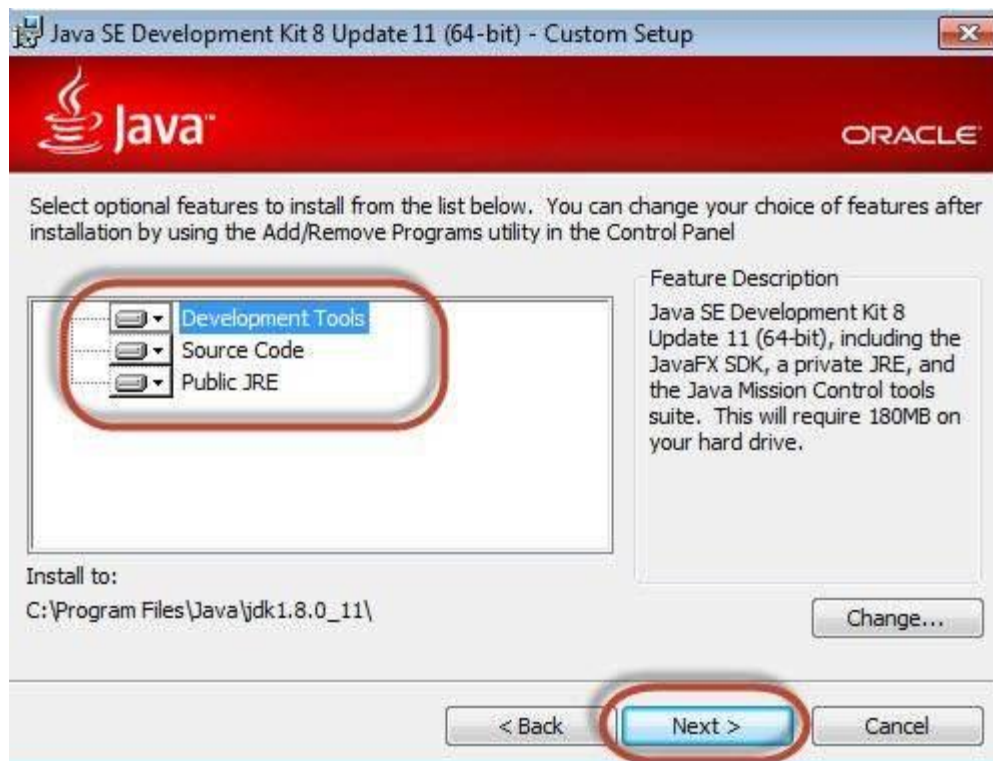
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	133.58 MB	 jdk-8u11-linux-i586.rpm
Linux x86	152.55 MB	 jdk-8u11-linux-i586.tar.gz
Linux x64	133.89 MB	 jdk-8u11-linux-x64.rpm
Linux x64	151.65 MB	 jdk-8u11-linux-x64.tar.gz
Mac OS X x64	207.82 MB	 jdk-8u11-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	135.66 MB	 jdk-8u11-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	96.14 MB	 jdk-8u11-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	135.7 MB	 jdk-8u11-solaris-x64.tar.Z
Solaris x64	93.18 MB	 jdk-8u11-solaris-x64.tar.gz
Windows x86	151.81 MB	 jdk-8u11-windows-i586.exe
Windows x64	155.29 MB	 jdk-8u11-windows-x64.exe

**Step 5 :** Run the downloaded exe file to launch the Installer wizard. Click 'Next' to continue.

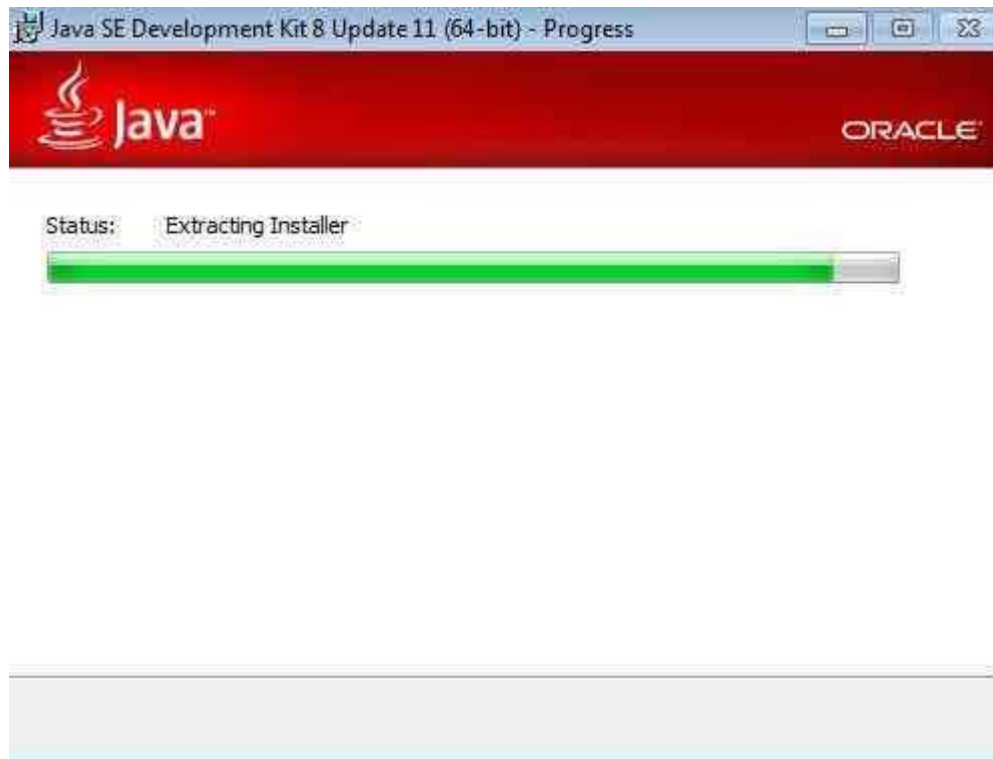


**Step 6 :** Select the features and click 'Next'.



**Step 7 :** The installer is extracted and its progress is shown in the wizard.





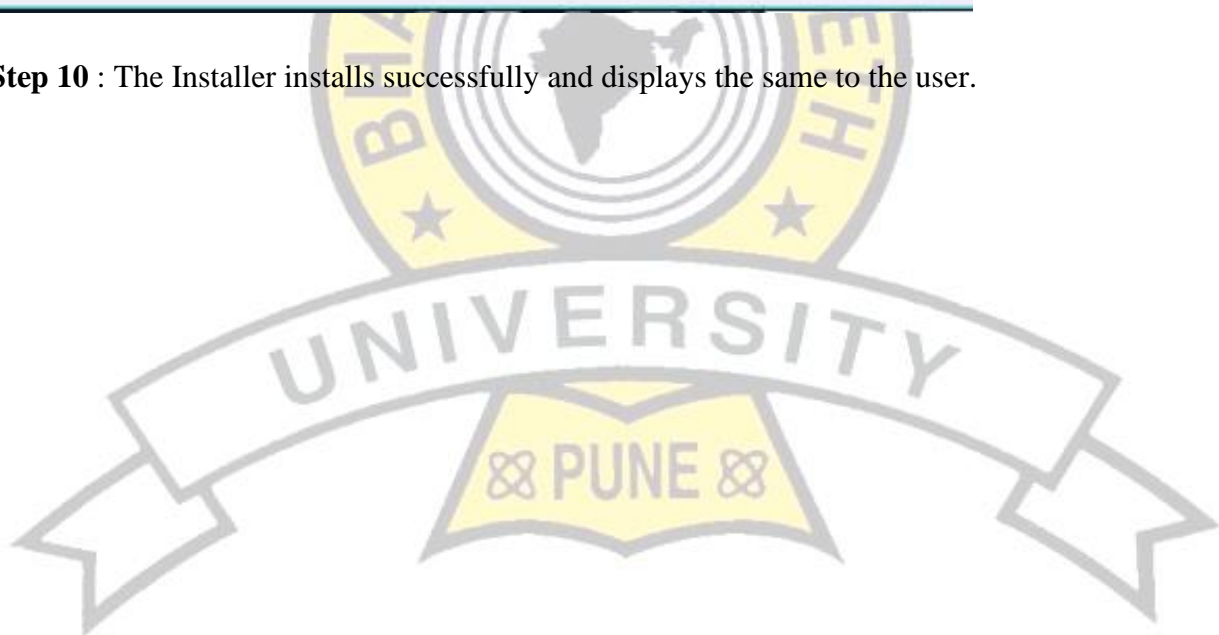
**Step 8 :** The user can choose the install location and click 'Next'.



**Step 9 :** The installer installs the JDK and new files are copied across.



**Step 10 :** The Installer installs successfully and displays the same to the user.





**Step 11 :** To verify if the installation was successful, go to the command prompt and just type 'java' as a command. The output of the command is shown below. If the Java installation is unsuccessful or if it had NOT been installed, it would throw an "unknown command" error.

```
C:\Windows\system32\cmd.exe

G:\>java
Usage: java [-options] class [args...]
           (to execute a class)
 or java [-options] -jar jarfile [args...]
           (to execute a jar file)
where options include:
    -d32          use a 32-bit data model if available
    -d64          use a 64-bit data model if available
    -server       to select the "server" VM
                  The default VM is server.

    -cp <class search path of directories and zip/jar files>
    -classpath <class search path of directories and zip/jar files>
                  A ; separated list of directories, JAR archives,
                  and ZIP archives to search for class files.
    -D<name>=<value>
                  set a system property
    -verbose:[class|gc|jni]
                  enable verbose output
    -version      print product version and exit
    -version:<value>
                  require the specified version to run
    -showversion  print product version and continue
    -jre-restrict-search | -no-jre-restrict-search
                  include/exclude user private JREs in the version search
    -? -help      print this help message
    -X            print help on non-standard options
    -ea[:<packagename>...![:<classname>]]
    -enableassertions[:<packagename>...![:<classname>]]
                  enable assertions with specified granularity
    -da[:<packagename>...![:<classname>]]
    -disableassertions[:<packagename>...![:<classname>]]
                  disable assertions with specified granularity
    -esa | -enablesystemassertions
                  enable system assertions
    -dsa | -disablesystemassertions
                  disable system assertions
    -agentlib:<libname>[=<options>]
                  load native agent library <libname>, e.g. -agentlib:hprof
                  see also, -agentlib:jdwp=help and -agentlib:hprof=help
    -agentpath:<pathname>[=<options>]
                  load native agent library by full pathname
    -javaagent:<jarpath>[=<options>]
                  load Java programming language agent, see java.lang.instrument
    -splash:<imagepath>
                  show splash screen with specified image
See http://www.oracle.com/technetwork/java/javase/documentation/index.html for more details.
G:\>_
```

Download and Configure Eclipse

**Step 1:** Navigate to the URL: <http://www.eclipse.org/downloads/> and download the appropriate file based on your OS architecture.

[GETTING STARTED](#)[MEMBERS](#)[PROJECTS](#)[MORE ▾](#)[HOME](#) / [DOWNLOADS](#)[PACKAGES](#) / [JAVA™ 8 SUPPORT](#)Eclipse Luna (4.4) Release for Windows ▾**Eclipse Standard 4.4** 206 MBDownloaded 955,976 Times [Other Downloads](#)

Standard Eclipse package suited for Java and plug-in development plus adding new plugins; already includes Git, Marketplace Client, source code and...

[Windows 32 Bit](#)  
[Windows 64 Bit](#)

**Step 2 :** Click the 'Download' button.

[GETTING STARTED](#)[MEMBERS](#)[PROJECTS](#)[MORE ▾](#)[HOME](#) / [DOWNLOADS](#) / [ECLIPSE DOWNLOADS - MIRROR SELECTION](#)[Downloads Home](#)[Source code](#)[More Packages](#)

## Eclipse downloads - mirror selection

All downloads are provided under the terms and conditions of the [Eclipse Foundation Software User Agreement](#) unless otherwise specified.

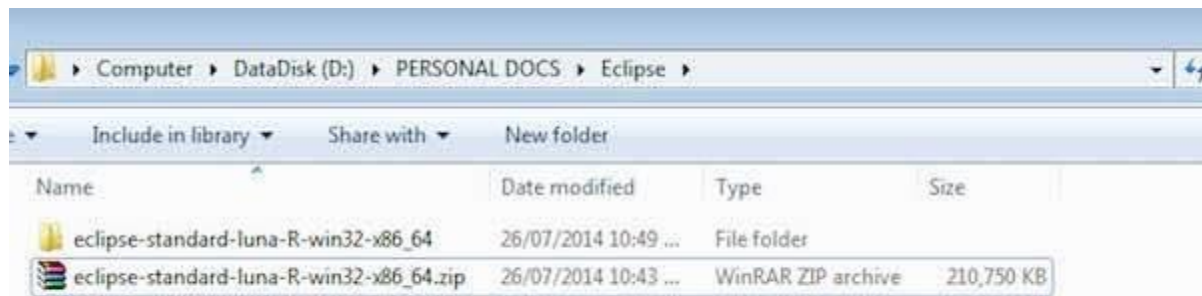
**Download eclipse-standard-luna-R-win32-x86\_64.zip from:**

Give Back to  
Eclipse

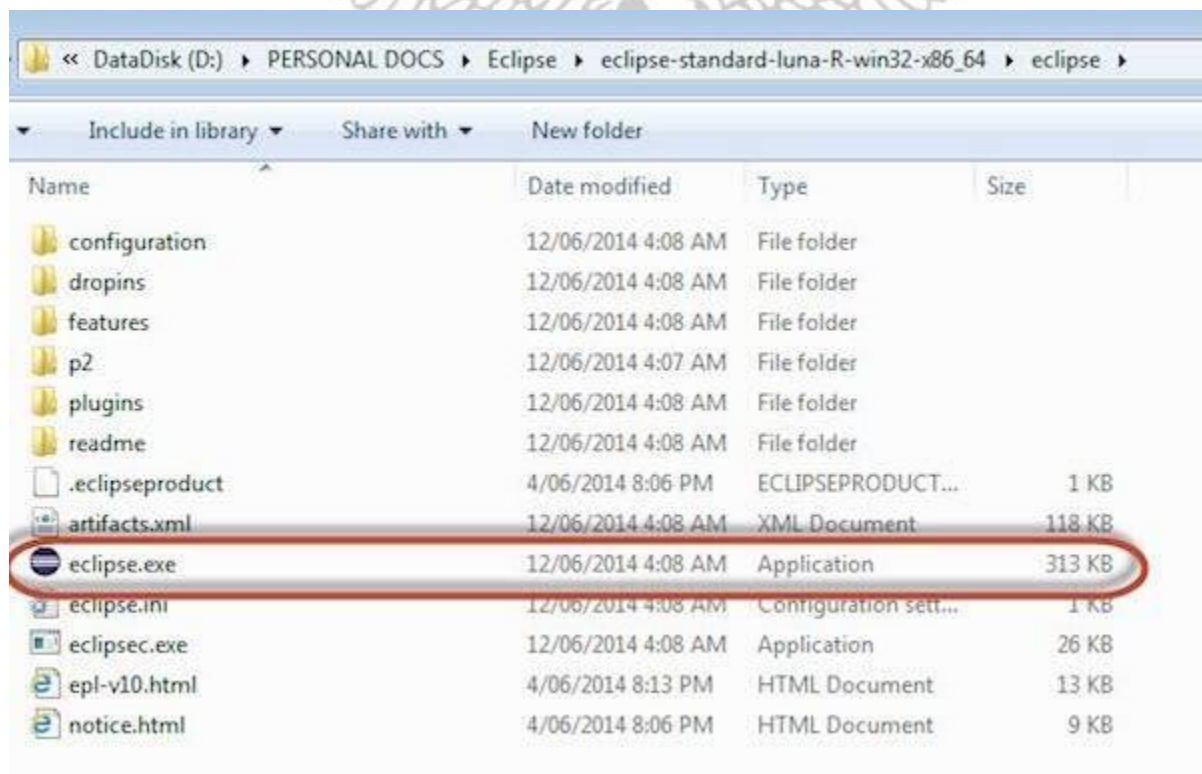
[\\$5](#)[\\$15](#)[\[China\] Beijing Institute of Technology \(http\)](#)Checksums: [\[MD5\]](#) [\[SHA1\]](#) [\[SHA-512\]](#)

...or pick a mirror site below.

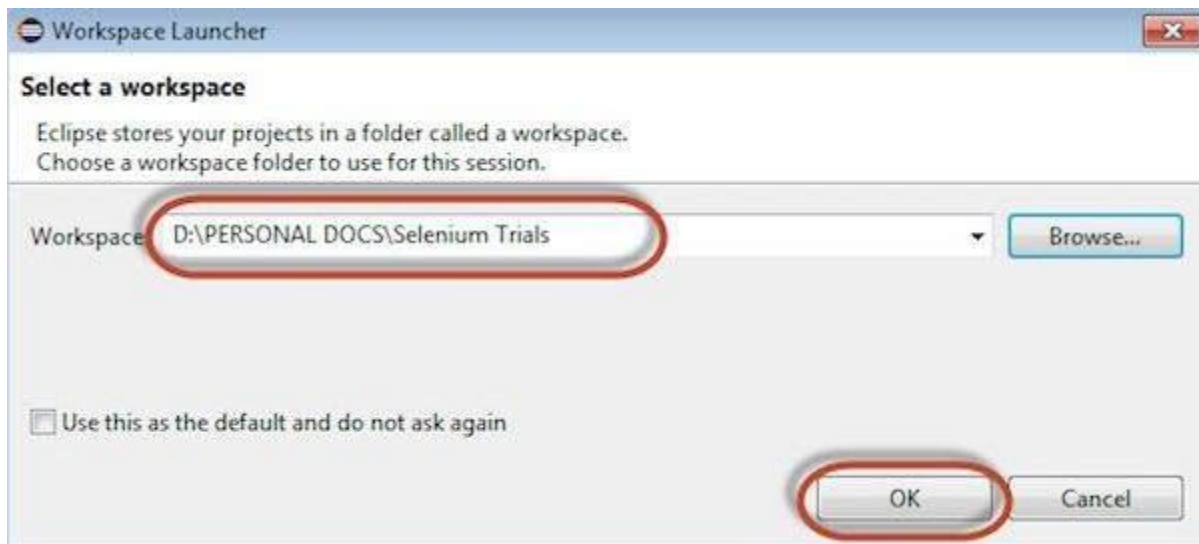
**Step 3 :** The download would be in a Zipped format. Unzip the contents.



**Step 4 :** Locate Eclipse.exe and double click on the file.



**Step 5 :** To configure the workspace, select the location where the development has to take place.



**Step 6 :** The Eclipse window opens as shown below.



Configure FireBug and FirePath

To work with Selenium RC or WebDriver, we need to locate elements based on their XPath or ID or name, etc. In order to locate an element, we need tools/plugins.

**Step 1 :** Navigate to the URL : <https://addons.mozilla.org/en-US/firefox/addon/firebug/> and download plugin.

**Selenium - Test design techniques**

There are various components involved in designing the tests. Let us understand some of the important components involved in designing a framework as well. We will learn the following topics in this chapter:

- Page Object Model
- Parameterizing using Excel
- Log4j Logging
- Exception Handling
- Multi Browser Testing
- Capture Screenshots
- Capture Videos



## **Experiment No:6**

### **Software Testing Roles and Responsibilities**

In case of software testing every company defines its own level of hierarchy, roles and responsibilities but on a broader level, if you take a look you will always find the following two levels in a software testing team:

**Test lead/manager:** A test lead is responsible for:

Defining the testing activities for subordinates – testers or test engineers.

All responsibilities of test planning.

To check if the team has all the necessary resources to execute the testing activities.

To check if testing is going hand in hand with the software development in all phases.

Prepare the status report of testing activities.

Required Interactions with customers.

Updating project manager regularly about the progress of testing activities.

Test engineers/QA testers/QC testers are responsible for:

To read all the documents and understand what needs to be tested.

Based on the information procured in the above step decide how it is to be tested.

Inform the test lead about what all resources will be required for software testing.

Develop test cases and prioritize testing activities.

Execute all the test case and report defects, define severity and priority for each defect.

Carry out regression testing every time when changes are made to the code to fix defects.

### **Overview of Software Engineering Team:**

How a software application shapes up during the development process entirely depends on the how the software engineering team organizes work and implements various methodologies. For an application to develop properly, it is important that all processes incorporated during the software development are stable and sustainable. Many times developers come under pressure as the delivery date approaches closer this often affects the quality of the software. Rushing through the processes to finish the project on time will only produce a software application which has no or minimal use for the customers. Hence, work organization and planning is important and sticking to the plan is very important. The project manager should ensure that there are no obstacles in the development process and if at all there is an issue it must be resolved with immediate attention.

It is very important to ensure that the software testing team has a proper structure. The hierarchy and roles should be clearly defined and responsibilities too should be well defined and properly distributed amongst the team members. When the team is well organized the work can be handled well. If every team member knows what duties he or she has to perform then they will be able to finish their duties as required well within the time limit. It is important to keep track of the testers' performance. It is very important to check what kind of defects the tester is able to uncover and what kind of defects he tends to miss. This will give you a fair idea about how serious your team is about the work.

All the team members should work together to prepare a document that clearly defines the roles and responsibilities of all the team members. Once the document is prepared the role of each member should be communicated clearly to everyone. Once the team members are clear about who is going to handle which area of the project, then in case of any issue it will be easy to determine who needs to be contacted.

Each member of the team should be provided with the necessary documents that provide information on how the task would be organized, what approach will be followed, how things are

scheduled, how many hours have been allocated to each member and all details related to applicable standards and quality processes.

### **Software Tester Role:**

A Software tester (software test engineer) should be capable of designing test suites and should have the ability to understand usability issues. Such a tester is expected to have sound knowledge of software test design and test execution methodologies. It is very important for a software tester to have great communication skills so that he can interact with the development team efficiently. The roles and responsibilities for a usability software tester are as follows:

A Software Tester is responsible for designing testing scenarios for usability testing.

He is responsible for conducting the testing, thereafter analyze the results and then submit his observations to the development team.

He may have to interact with the clients to better understand the product requirements or in case the design requires any kind of modifications.

Software Testers are often responsible for creating test-product documentation and also has to participate in testing related walk through.

A software tester has different sets of roles and responsibilities. He should have in depth knowledge about software testing. He should have a good understanding about the system which means technical (GUI or non-GUI human interactions) as well as functional product aspects. In order to create test cases it is important that the software tester is aware of various testing techniques and which approach is best for a particular system. He should know what are various phases of software testing and how testing should be carried out in each phase. The responsibilities of the software tester include:

Creation of test designs, test processes, test cases and test data.

Carry out testing as per the defined procedures.

Participate in walkthroughs of testing procedures.

Prepare all reports related to software testing carried out.

Ensure that all tested related work is carried out as per the defined standards and procedures.

### **Software Test Manager Role:**

Managing or leading a test team is not an easy job. The company expects the test manager to know testing methodologies in detail. A test manager has to take very important decisions regarding the testing environment that is required, how information flow would be managed and how testing procedure would go hand in hand with development. He should have sound knowledge about both manual as well as automated testing so that he can decide how both the methodologies can be put together to test the software. A test manager should have sound knowledge about the business area and the client's requirement, based on that he should be able to design a test strategy, test goal and objectives. He should be good at project planning, task and people coordination, and he should be familiar with various types of testing tools. Many people get confused between the roles and responsibilities of a test manager and test lead. For a clarification, a test lead is supposed to have a rich technical experience which includes, programming, handling database technologies and various operating systems, whereas he may not be as strong as Software Test Manager regarding test project management and coordination. The responsibilities of the test manager are as follows:

Since the test manager represents the team he is responsible for all interdepartmental meetings.

Interaction with the customers whenever required.

A test manager is responsible for recruiting software testing staff. He has to supervise all testing activities carried out by the team and identify team members who require more training. Schedule

testing activities, create budget for testing and prepare test effort estimations. Selection of right test tools after interacting with the vendors. Integration of testing and development activities.

Carry out continuous test process improvement with the help of metrics.

Check the quality of requirements, how well they are defined.

Trace test procedures with the help of test traceability matrix.

### **Software Test Automator Role:**

Software test automator or an automated test engineer should have very good understanding of what he needs to test- GUI designs, load or stress testing. He should be proficient in automation of software testing, and he should be able to design test suites accordingly. A software test automator should be comfortable using various kinds of automation tools and should be capable of upgrading their skills with changing trends. He should also have programming skills so that he is able to write test scripts without any issues. The responsibilities of a tester at this position are as follows:

He should be able to understand the requirement and design test procedures and test cases for automated software testing. Design automated test scripts that are reusable.

Ensure that all automated testing related activities are carried out as per the standards defined by the company.

### **Interactions between Software Test Team And Business Teams**

If at all a customer has any issues related to testing activities and operational matters of the project then it is the software testing manager who is responsible for communicating the details to the client regarding how things are being managed. The software testing manager not only answers the queries of the customers but also ensures that the project is completed on time as per the requirement of the customer.

### **Interactions between Software Test Team And Development Teams**

In order to produce good software applications, it is important that software testing and software development teams work together with good understanding. For this it is important that the testers and developers are comfortable with each other's role and understand well that they have a common goal and it is wise to listen each other. A good communication skill is very important both for testers and developers.

Before getting started with testing work it is important to discuss the basic guidelines and expectations so that there is no confusion in later stages. Criticism should be taken in a positive sense. It is important to understand that developers and testers have a common goal of producing high quality software. A tester is not discovering bugs to show someone down, the idea is to learn from mistakes and avoid repeating them in future. A culture of constructive criticism can be of great help.

### **Interactions between Software Test Team And Release Management Teams**

The release management teams are responsible for moving the software from development into production. This team is responsible for planning the releases for hardware, software and testing. It is also responsible for development of software development procedures and for coordinating interactions and training of releases. Software testing is considered to be a very important aspect of software engineering life cycle but it does not get over with development. Testing and verification is a very important part of release management exercise.

### **Interactions between Software Test Manager And Software Project Manager**

The job of a software test manager is not an easy one. He has to recruit testing team and take responsibility for getting them trained. A software manager has to perform ongoing analysis of various testing processes and ensure that the testing team is carrying out all the processes correctly. This job is of great responsibility as the software testing manager is the one who selects, introduces and implement various tools for testing. A software test manager is responsible for finalizing templates for testing documents, test reports and other procedures.

Since a software tester manager has to deal with all the details of various testing activities, it is very important for him to be in constant touch with the project manager and provide necessary support in project planning and scheduling so that the project can be successfully completed in time within the specified financial budget limits.



## **Experiment No:7**

### **Study of any bug tracking tool (e.g. Bugzilla, bugbit)**

#### **Introduction:**

Bugzilla is the most popular bug tracking system available today. Bugzilla's popularity is due to its highly customizable interface, easy configuration, and a large community of very active users. With Bugzilla, you can begin configuring your products, users, and bugs within minutes of starting up the system. Bugzilla provides you with many features, including:

- LDAP integration
- SMTP and Sendmail support
- Internationalized
- User authentication, group security, and SSL support
- Voting system
- Shortcut flags
- Custom fields
- Keyword tagging
- Saved searches
- Voting module
- Multiple report types

In this experiment, you'll learn how to:

- Set parameters and default preferences.
- Create new users.
- Create products and components.
- Modifying default field values.
- Creating new bugs and modifying existing bugs.

- **Setting Parameters and Default Preferences.**

When you start using Bugzilla, you'll need to set a small number of parameters and preferences. At a minimum, you should change the following items, to suit your particular need:

- Set the **maintainer**
- Set the **mail\_delivery\_method**
- Set **bug change policies**
- 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 you're using. If you're evaluating a click2try application, select **Test**. If you're using SMTP, set any of the other SMTP options for your environment.
6. Click **Save Changes**.
7. In the left side **Index** list, click **Bug Change Policies**.
8. Select On for **commentoncreate**, which will force anyone who enters a new bug to enter a comment, to describe the bug.
9. Click **Save Changes**.
10. Click **Default Preferences** at the bottom of the page.
11. Select the display order from the drop-down list next to the **When viewing a bug, show comments in this order** field.
12. Click **Submit Changes**.

- **Creating a New User**

Before you begin entering bugs, make sure you add some new users. You 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 **add** 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**. You'll probably want to enable the following options in the row titled User is a member of these groups:
  - o **canconfirm**
  - o **editbugs**
  - o **editcomponents**

- **Adding Products**

You'll add a product in Bugzilla for every product you are developing. To start with, when you first login to Bugzilla, you'll find a test product called **TestProduct**. You 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 telling you that you’ll need to add at least one component.

**To add a component:**

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

- **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 you want to edit. The OS field contains a list of operating system names. You are going to add browsers to this list. In reality, you 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**.
4. In the **Value** field, enter “IE7.”
5. Click **Add**.
6. Click **Add a value** again.
7. In the **Value** field, enter “Firefox 3.”
8. Click **Add**.
9. Where it reads **Add other values for the op\_sys field**, click **op\_sys**. This redisplay the table. You 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 you create a new bug.

- **To create a new bug:**

1. In the top menu, click **New**.
2. If you'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** drop-down list to see the options, including the new browser names you entered.
5. Select one of the options.
6. Enter a summary and a description. You can add any other information you like, but it is not required by the system, although you may determine that your bug reporting policy requires certain information.
7. Click **Commit**. Bugzilla adds your bug report to the database and displays the detail page for that bug.

**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 you think might be in the title of the bug.
6. Click **Search**. If any bugs meet the criteria you entered, Bugzilla displays them in a list summary.
7. Click the **ID** number link to view the full bug report.

- **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. Remember it must be in the format of an email address.
3. Click **Commit**.

## **Experiment No:8**

### **Study of any test management tool (e.g. Test Director)**

The Test Director testing process includes four phases:

- Specifying Requirements
- Planning Tests
- Running Tests
- Tracking Defects

#### **Starting Test Director:**

You start TestDirector from your Web browser, using the TestDirector URL.

#### **To start TestDirector:**

##### **1 Open the TestDirector Options window.**

In your Web browser, type your TestDirector URL:

<http://<TestDirector server name>/<virtual directory name>/default.htm>

The TestDirector Options window opens.

##### **2 Open TestDirector.**

Click the **TestDirector** link.

The first time you run TestDirector, the application is downloaded to your computer. Then, each time you open TestDirector, it automatically carries out a version check. If TestDirector detects a newer version, it downloads the latest version to your machine.

The TestDirector Login window opens.

## Select a domain.

In the **Domain** list, select **DEFAULT**.

## Select a project.

In the **Project** list, select **TestDirector\_Demo**.

## Log on to the project as a QA tester.

In the **User ID** box, type one of the following user names: **alice\_td**, **cecil\_td**, or **michael\_td**. Skip the **Password** box. A password was not assigned to any of the above user names. Click the **Login** button.

TestDirector opens and displays the module in which you were last working. In the title bar, TestDirector displays the project name and your user name.

## To explore the TestDirector window:

### 1 Explore the TestDirector modules.

- Click the **Requirements** tab. The Requirements module enables you to specify your testing requirements. This includes defining what you are testing, defining requirement topics and items, and analyzing the requirements.
- Click the **Test Plan** tab. The Test Plan module enables you to develop a test plan based on your testing requirements. This includes defining goals and strategies, dividing your plan into categories, developing tests, automating tests where beneficial, and analyzing the plan.
- Click the **Test Lab** tab. The Test Lab module enables you to run tests on your application and analyze the results.
- Click the **Defects** tab. The Defects module enables you to add defects, determine repair priorities, repair open defects, and analyze the data.

## Specifying Testing Requirements

- Defining Requirements
- Viewing Requirements
- Modifying Requirements
- Converting Requirements

## Planning Tests

- Developing a Test Plan Tree
- Designing Test Steps
- Copying Test Steps
- Calling Tests with Parameters
- Creating and Viewing Requirements Coverage
- Generating Automated Test Scripts

## Running Tests

- Defining Test Sets
- Adding Tests to a Test Set
- Scheduling Test Runs
- Running Tests Manually
- Running Tests Automatically

## Adding and Tracking Defects

- Adding New Defects
- Matching Defects
- Updating Defects
- Mailing Defects
- Associating Defects with Tests
- Creating Favorite Views\

## **Experiment No:9**

### **Study of any open source-testing tool (e.g. Test Link)**

**Test Link:** 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.

Its 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 reports.

Test link exposes API, written in PHP, can help generate quality assurance dashboards. The functions like AddTestCaseToTestPlan, AssignRequirements,CreateTestCase etc. helps create and organize test cases per test plan. Functions like GetTestCasesForTestPlan, GetLastExecutionResult allows one to create quality assurance dashboard.

#### **How to use Test Link: Example of TestLink simple work-flow:**

1. Initial step would be to create a new Test Project and assign QA testers or engineers with tasks.
- 2.Import Software Requirements and for part of these requirements generates empty Test Cases. Reorganize them into Test Suites.
- 3.Create a content for empty test cases using test specifications that are being organized into Test suites.
- 4.Create “Regression testing” and assigns to applicable test cases.
- 5.Create a Test Plan, Build and link all Test Cases in Test Suite to this Test Plan. Assign resources to this test plan.
- 6.Assume QA got there first Build or Release Candidate from development team, execute and record the testing with the result.
- 7.Assume QA get new Build or Release Candidate with fixes for blocking issues, verify these fixes for

blocking issues, execute regression tests.

8. Manager (Test or Engineering) and other project related stakeholders want to see results and status of testing. Then in such a case, these stakeholders including managers can create accounts or use default Guest account to view test results for a particular Build. An overall report gets generated for automated test suites, as a Guest manager is able to view test results at a higher level in graphical format.

9. Suppose new changes happen to existing functionality, it's very easy to modify existing test plan and add new test cases or enhance/modify existing test cases and attach them to a particular test plan.

10. Test suites continue to execute as usual by generating various reports.

11. For new project again QA creates a new Test, follows above steps to implement TestLink for there project.



## **Experiment No:10**

### **Case Study on Software Testing**

**This experiment is based on any real world case study in the field of software testing along with documentation.**



