MANUAL MODULE 2

Session 1

MANAGEMENT

There are different Types of management:

- Requirement Management (RTM)
- Risk Management
- Change Management
- Configuration Management
- PMLC(Product Management Life cycle)
- Project Management

RTM (Requirement Traceability Matrix)

Requirements can be managed manually/automation.For managing manually use RTM .

Eg: Requisite PRO

RTM is a tool used to identify and track requirements throughout a project life cycle. It can be represented in the form of a table,showing many to many relationships with requirements and test cases. RTM shows that in every case it shows many relationships, mainly 3 types of traceability.

1.Forward RTM -it means we are moving from the earlier stage of development to later stage of development.That mapping takes place from requirement to end product is known as forward RTM.

2.Backward RTM - Mapping takes place from end product to requirement is backward RTM.

3.Bidirectional RTM - Using both forward & backward traceability is called bidirectional RTM.

Change Management

Software changes are requirements that may be introduced into a project that may be introduced into a project as it evolves.when modification to existing requirements or requested or when business needs are redefined.Incorporating effective change control procedures ensure that requirement change requests are implemented in an accurate and timely manner.

i)Scope creep is what happens when changes are made to the project scope without any control procedure like change requests. Those changes also affect the project schedule, budget, costs, resource allocation and might compromise the completion of milestones and goals. Scope creep is one of the most common project management risks.

ii)Version Control: It is required during the development process &after implementation problem occur for testers when the version being tested is not the same version as that which development team has completed.

Configuration Management

It refers to the management of all components of a system including hardware,OS,network.It also involves managing the interaction of all configuration components whenever a change is made.

PMLC (Product Management Life Cycle)

It is the process of managing the entire life cycle of a product.PMLC have core team.This core team made up of one representatives from each department.Marketing project management.QA and development team are responsible for monitoring.The process and verifying at each phase that all deliverable have been completed before moving to the next phase.

PMLC consists of different phases.

● Requirement Specification - The marketing department or project management team authors the initial draft(outline of the project) and revision of the requirement document.

A team as described in the inspection section of this document performs the actual inspection.

Input

- ✓ Interview with existing customer
- ✓ Information from previous product
- ✓ Information from other departments.
- ✓ New business requirement

Output

- ✓ Inspected requirement document.

● Product Specification - The marketing department,project management team and development staff authors the initial trial and revision of the project specification document.It contains user interface description of the function described in the requirement document.

Input

- ✓ Inspected requirements document
- ✓ Information from previous product
- ✓ Information from other departments.
- ✓ QA begins writing the test plan.

Output

- ✓ Inspected product specification document.

● Functional Specification & Test Plan - The research & development department authors the initial draft & revision of the final specification.it

contains user interface description of the function described in the product specification document.

Input

✓ Inspected product specification document

Output

✓ Inspected functional document

Test Plan -Initial & revision of the test plan are authorised by the QA department.It describes the tests to be performed.The resources that are needed & list the test cases for all features of the software to be tested.

Input

✓ Inspected product specification document

Output

✓ Inspected test plan document

✓ Computed test plan

● Coding & Test cases/scripts - when each module is completed,unit test plan written,afterwards the source code & unit test plan for the modules go through a walkthrough or inspection.

Input

✓ Inspected product specification document

Output

✓ Inspected test cases or script

● General Release - Production is when software is actually shipped to customers for real world use.

● Maintenance - enhancement are no longer considered & only existing problems are corrected.

Risk Management

Risk---> The probability of a negative event occur.The potential loss is accurate with that event.The first part of risk management is to understand,identify and determine the magnitude of risks.

Two activities associated with risk management.

● Risk Reduction Method

Tools or methods to avoid the occurrence of risk.First quantify the risk.The formula to quantify risk to multiply the frequency of an undesirable occurrence times the loss associated with that occurrence.

● Contingency Planning

It is a process that prepares an organisation to respond coherently to an unplanned event.

Session 2

BUILDING OF TEST POLICY

Test bed

An environment containing hardware,software tools,simulators,instrumentation and other supporting elements to conduct test.

Use Case

Describe how a user uses a system to achieve a goal.it provides a format for capturing technical requirements applied to system release.

Use Case diagram has 3 components:

1 .Actor

2. Use Case

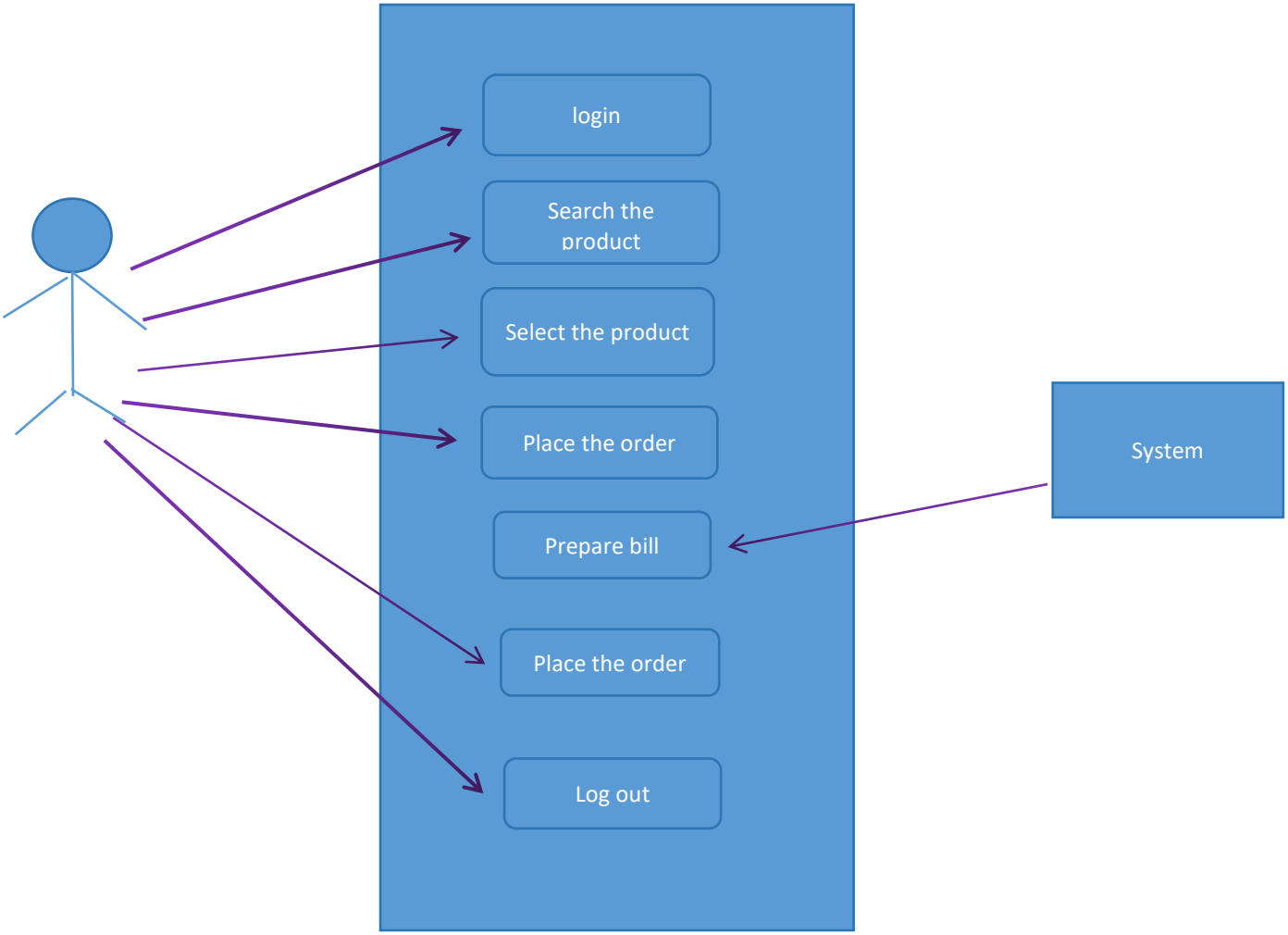
3.System boundary

Actor:- A role that the user plays with respect to the system including human uses and other systems.

Use Case:- A set of scenarios that describing an interaction between user and system.

System Boundary:- rectangular diagram representing the system boundary between actors and the system.

Eg: Online shopping



Test policy

Some standards & procedures to achieve a task.A testing policy is the management's objective for testing. It is the objective to be accomplished.A process must be in place to determine how that policy will be achieved.

Test Strategy: Test strategy is a set of guidelines that explains test design and determines how testing needs to be done .

Prerequisites of test planning

1. Test objectives

- Goal :what are the factors to be tested
- Meeting requirements

2. Acceptance criteria

To reduce the communication gap between user and organization.

3. Assumption

When the testing should start?

4. People issue

It may be political and personal.The people issue includes who should run the project,who can make decisions etc.

5. Constraints

Constraints are test staff size,test schedule and budget.other constraints can include,inability to access user database for test purposes,limited access to hardware facilities for test purpose.

Session 3

LEVELS OF TESTING

1. Unit Testing :-Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.A unit is a single testable part of a software system and tested during the development phase of the application software.The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.

2. Integration testing is the second level of the software testing process after unit testing. In this testing, units or individual components of the software are tested in a group. The focus of the integration testing level is to expose defects at the time of interaction between integrated components or units.Unit testing uses modules for testing purpose, and these modules are combined and tested in integration testing. The Software is developed with a number of software modules that are coded by different coders or programmers. The goal of integration testing is to check the correctness of communication among all the modules.

Big Bang

All components or system are integrated simultaneously after which everything is tested as a whole.

Incremental Testing

In this all programs are integrated one by one and combine two or more modules then a test is carried out.

- ✧ **Top down**

Begins testing from the top of the modules hierarchy.

- ✧ **Bottom up**

Begins testing from bottom of the hierarchy and works up to the top.

3. System Testing : System Testing includes testing of a fully integrated software system. Generally, a computer system is made with the integration of software (any software is only a single element of a computer system). The software is developed in units and then interfaced with other software and hardware to create a complete computer system. In other words, a computer system consists of a group of software to perform the various tasks, but only software cannot perform the task; for that software must be interfaced with compatible hardware. System testing is a series of different types of tests with the purpose to exercise

and examine the full working of an integrated software computer system against requirements.

4. User Acceptance Testing : Building the confidence of the client and user is the role of acceptance test phase.it depends on business scenarios. UAT can be done in 2 ways:

- Alpha Testing
- Beta Testing

Session 4

STATIC TESTING

Static testing is testing, which checks the application without executing the code. It is a verification process. Some of the essential activities are done under static testing such as business requirement review, design review, code walkthroughs, and the test documentation review. Static testing is performed in the white box testing phase, where the programmer checks every line of the code before handing it over to the Test Engineer. Static testing can be done manually or with the help of tools to improve the quality of the application by finding the error at the early stage of development; that is why it is also called the verification process.

Why do we need to perform Static Testing?

- ❖ We can perform static testing to fulfill the below aspects:
- ❖ We can use static testing to improve the development productivity.
- ❖ If we performed static testing on an application, we could find the detects in the earlier stages and easily fix them.
- ❖ The usage of static testing will decrease the testing cost, development timescales, and time.

Objectives of Static testing

The main objectives of performing static testing is as below:

- Static testing will decrease the flaws in production.
- Static testing will identify, anticipate and fix the bugs at the earliest possible time.
- It is used to save both time and cost.
- It is used to identify defects in the early stage of SDLC, where we can fix them easily.

Advantages of Static Testing

- Improved Product quality
- Static testing will enhance the product quality because it identifies the flaws or bugs in the initial stage of software development.
- Improved the efficiency of Dynamic testing
- The usage of Static testing will improve Dynamic Testing efficiency because the code gets cleaner and better after executing Static Testing.
- As we understood above, static Testing needs some efforts and time to generate and keep good quality test cases.
- Reduced SDLC cost
- The Static Testing reduced the SDLC cost because it identifies the bugs in the earlier stages of the software development life cycle. So, it needs less hard work and time to change the product and fix them.
- Immediate evaluation & feedback
- The static testing provides us immediate evaluation and feedback of the software during each phase while developing the software product.
- Exact location of bug is traced
- When we perform the static testing, we can easily identify the bugs' exact location compared to the dynamic Testing.

Static testing techniques

Testing Review

A review in a static testing is a process or meeting conducted to find the potential defects in the design of any program.

- ✧ Moderator :performs entry check,follow up on rework,coaching team member,scheduling meeting.
- ✧ Author :Take responsibility for fixing the defect found and improves the quality of the document.
- ✧ Scribe: It does the logging of the defect during a review and attends the review meeting.
- ✧ Reviewer:Check material for defect and inspects.
- ✧ Manager:Decide on the execution of reviews and ensures the review process objectives are met.

Peer Review

A peer review,a review technique which is a static white box testing which are conducted to spot the defects early in the life cycle that cannot be detected by black box testing technique.

Formal Review

It is the one of the most important review technique used in static testing.Conducted by a group of 3 or more individuals.The main objective of the formal review is to be evaluate software confirmation with specification and plan as well as to ensure the change integrity.

Activities :

- ✧ Planning -The initial stage of the whole review process.Planning is extremely important.It is during this stage that request to review the software and its components is presented by the author to the moderator.
- ✧ Kick-off -this is the second stage of the formal review process,where the leader defines the main objective and goal of review to the whole team.

- ✧ Individual Preparation- As the name suggests,here the reviewers review the document individually based on the provided checklists,document,rules and procedures.
- ✧ Review Meeting -This usually involves 3 phases:Logging,discussion,decision where in different task related to the document under review is performed.
  - ■ Logging:This issues identified in the preparation stage are logged here by the author or scribe.These issues are further categorized according to their severity:critical,major & minor.
  - ■ Discussion :The review provided at this stages are found to be cost effective,as they are identified at the earlier stage,as the cost of rectifying a defect in a earlier stages would be much more than doing it in the initial stages.
  - ■ Decision : After the commencement of review meeting,the team is tasked to make a decision regarding the document under review which can be based on exit criteria.
- ✧ Rework

If the number of defects found are more than an expected level,then document has to be reworked.

- ✧ Follow up

During this stage of formal review process,the moderator makes sure that the author take care of all the defects.

Informal Reviews

A two person team can conduct an informal review.The goal is to keep the author and to improve the quality of the document.

Inspection

It is usually led by a trained moderator.the document under inspection is prepared and checked thoroughly by the reviewer before

meeting,comparing the work product with its source and other referenced documents and using rules and checklists.

Walkthrough

Walkthrough is a method of conducting informal group/individual review. In a walkthrough, author describes and explain work product in a informal meeting to his peers or supervisor to get feedback. Here, validity of the proposed solution for work product is checked.

It is cheaper to make changes when design is on the paper rather than at time of conversion. Walkthrough is a static method of quality assurance. Walkthrough are informal meetings but with purpose.

Session 5

DYNAMIC TESTING

Dynamic testing is testing, which is done when the code is executed at the runtime environment. It is a validation process are where functional testing [unit, integration, and system testing] and non-functional testing [user acceptance testing performed. We will perform dynamic testing to check whether the application or software is working fine during and after the installation of the application without any error.

Why do we need to perform Dynamic Testing?

- ✧ We will perform dynamic testing to check whether the application or software is working fine during and after installing the application without any error.
- ✧ We can perform dynamic testing to verify the efficient behavior of the software.
- ✧ The software should be compiled and run if we want to perform dynamic testing.

- ✧ Generally, Dynamic Testing is implemented to define the dynamic behavior of code.
- ✧ The team implements the code to test the software application's performance in a run-time environment during the dynamic testing process.

Characteristic of Dynamic Testing

- ✧ It is implemented throughout the **validation stage** of software testing.
- ✧ Dynamic Testing is done by performing the program.
- ✧ Both functional and non-functional testing include in dynamic testing.
- ✧ In Dynamic testing, we can easily identify the bugs for the particular software.
- ✧ It helps the team in validating the reliability of the software application.

Types of Dynamic testing

Dynamic testing divided into two different testing approach, which are as follows:

- **White-box testing**
- **Black-box testing**

Both the testing techniques will help us execute the dynamic testing process efficiently as they play an important role in verify the performance and quality of the software.

Advantages of Dynamic Testing

- ✧ It discloses very difficult and complex defects.
- ✧ It detects the defects that can’t be detected by static testing.
- ✧ It increases the quality of the software product or application being tested.

- ✧ Dynamic testing detects security threats and ensure the better secure application.
- ✧ It can be used to test the functionality of the software at the early stages of development.
- ✧ It is easy to implement and does not require any special tools or expertise.
- ✧ It can be used to test the software with different input values.
- ✧ It can be used to test the software with different data sets.
- ✧ It can be used to test the software with different user profiles.
- ✧ It can be used to test the functionality of the code.
- ✧ It can be used to test the performance of the code.
- ✧ It can be used to test the security of the code.

Disadvantages of Dynamic Testing

- It is a time consuming process as in dynamic testing whole code is executed.
- It increases the budget of the software as dynamic testing is costly.
- Dynamic testing may require more resources than static testing.
- Dynamic testing may be less effective than static testing in some cases.
- It is difficult to cover all the test scenarios.
- It is difficult to find out the root cause of the defects.

Difference between Static and Dynamic testing

Static Testing	Dynamic Testing
Testing was done without executing the program	Testing is done by executing the program
This testing does the verification process	Dynamic testing does the validation process
Static testing is about prevention of defects	Dynamic testing is about finding and fixing the

	defects
Static testing gives an assessment of code and documentation	Dynamic testing gives bugs/bottlenecks in the software system.
Static testing involves a checklist and process to be followed	Dynamic testing involves test cases for execution
This testing can be performed before compilation	Dynamic testing is performed after compilation

Session 6

WHITE BOX TESTING

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system.

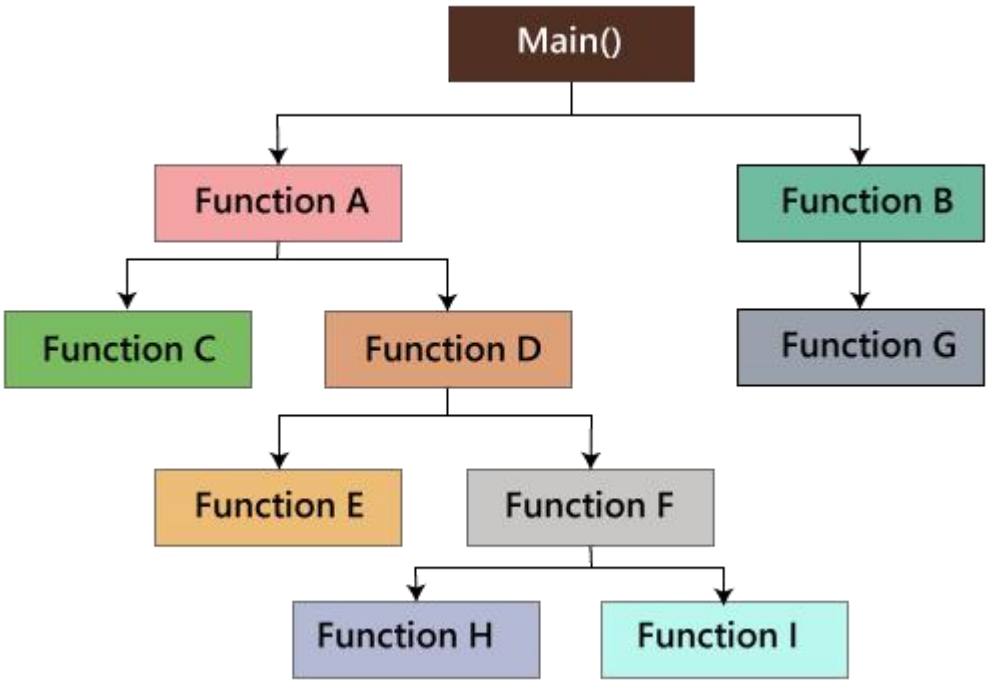
Various types of testing

The white box testing contains various tests, which are as follows:

- Path testing
- Loop testing
- Condition testing
- Testing based on the memory perspective
- Test performance of the program

Path testing

In the path testing, we will write the flow graphs and test all independent paths. Here writing the flow graph implies that flow graphs are representing the flow of the program and also show how every program is added with one another as we can see in the below image:



And test all the independent paths implies that suppose a path from main() to function G, first set the parameters and test if the program is correct in

that particular path, and in the same way test all other paths and fix the bugs.

Loop testing

In the loop testing, we will test the loops such as while, for, and do-while, etc. and also check for ending condition if working correctly and if the size of the conditions is enough.

For example: we have one program where the developers have given about 50,000 loops.

```
1.  {
2.  while(50,000)
3.  ……
4.  ……
5.  }
```

Condition testing

In this, we will test all logical conditions for both **true** and **false** values; that is, we will verify for both **if** and **else** condition.

For example:

```
1.  if(condition) - true
2.  {
3.  …..
4.  ……
5.  ……
```

```
6.      }
7.      else - false
8.      {
9.      …..
10.     ……
11.     ……
12.     }
```

The above program will work fine for both the conditions, which means that if the condition is accurate, and then else should be false and conversely.

Testing based on the memory (size) perspective

The size of the code is increasing for the following reasons:

- **The reuse of code is not there**: let us take one example, where we have four programs of the same application, and the first ten lines of the program are similar. We can write these ten lines as a discrete function, and it should be accessible by the above four programs as well. And also, if any bug is there, we can modify the line of code in the function rather than the entire code.
- The **developers use the logic** that might be modified. If one programmer writes code and the file size is up to 250kb, then another programmer could write a similar code using the different logic, and the file size is up to 100kb.
- The **developer declares so many functions and variables** that might never be used in any portion of the code. Therefore, the size of the program will increase.

For example,

```
1.    Int a=15;
2.    Int b=20;
3.    String S= "Welcome";
4.    ….
5.    …..
6.    …..
7.    ….
8.    …..
9.    Int p=b;
10.   Create user()
11.   {
12.   ……
13.   ……
14.   ….. 200's line of code
15.   }
```

Test the performance (Speed, response time) of the program

The application could be slow for the following reasons:

- When logic is used.
- For the conditional cases, we will use **or** & **and** adequately.
- Switch case, which means we cannot use **nested if**, instead of using a switch case.

Advantages of White box testing

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Techniques Used in White Box Testing

Data Flow Testing	Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.
Control Flow Testing	Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control

	graph of the program.
Branch Testing	Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.
Statement Testing	Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code.
Decision Testing	This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false.

Difference between white-box testing and black-box testing

Following are the significant differences between white box testing and black box testing:

White-box testing	Black box testing
The developers can perform white box testing.	The test engineers perform the black box testing.
To perform WBT, we should have an understanding of the programming languages.	To perform BBT, there is no need to have an understanding of the programming languages.

In this, we will look into the source code and test the logic of the code.	In this, we will verify the functionality of the application based on the requirement specification.
In this, the developer should know about the internal design of the code.	In this, there is no need to know about the internal design of the code.

Session 7

BLACK BOX TESTING

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

Techniques Used in Black Box Testing

Decision Table Technique	Decision Table Technique is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.
Boundary Value	Boundary Value Technique is used to test boundary values,

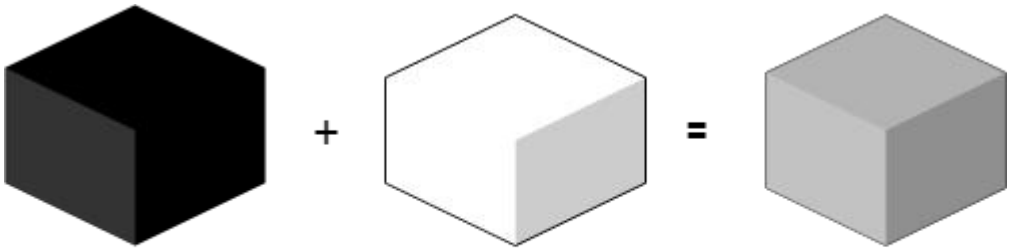
Technique	boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.
State Transition Technique	State Transition Technique is used to capture the behavior of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.
All-pair Testing Technique	All-pair testing Technique is used to test all the possible discrete combinations of values. This combination method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.
Cause-Effect Technique	Cause-Effect Technique underlines the relationship between a given result and all the factors affecting the result.It is based on a collection of requirements.
Equivalence Partitioning Technique	Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.
Error Guessing Technique	Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.
Use Case Technique	Use case Technique used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test scenario that

	can exercise the entire software based on the functionality of each function from start to end.

Session 8

GREY BOX TESTING

Grey box testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a **combination of black box and white box testing** because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.



Why Grey Box testing?

- It provides combined benefits of both Black box testing and White Box testing.
- It includes the input values of both developers and testers at the same time to improve the overall quality of the product.
- It reduces time consumption of long process of functional and non-functional testing.
- It gives sufficient time to the developer to fix the product defects.

- It includes user point of view rather than designer or tester point of view.

Techniques of Grey box Testing

Matrix Testing

This testing technique comes under Grey Box testing. It defines all the used variables of a particular program. In any program, variable are the elements through which values can travel inside the program. It should be as per requirement otherwise, it will reduce the readability of the program and speed of the software. Matrix technique is a method to remove unused and uninitialized variables by identifying used variables from the program.

Regression Testing

Regression testing is used to verify that modification in any part of software has not caused any adverse or unintended side effect in any other part of the software. During confirmation testing, any defect got fixed, and that part of software started working as intended, but there might be a possibility that fixed defect may have introduced a different defect somewhere else in the software. So, regression testing takes care of these type of defects by testing strategies like retest risky use cases, retest within a firewall, retest all, etc.

Orthogonal Array Testing or OAT

The purpose of this testing is to cover maximum code with minimum test cases. Test cases are designed in a way that can cover maximum code as well as GUI functions with a smaller number of test cases.

Pattern Testing

Pattern testing is applicable to such type of software that is developed by following the same pattern of previous software. In these type of software possibility to occur the same type of defects. Pattern testing determines reasons of the failure so they can be fixed in the next software.

Session 9

BRS vs SRS

The **BRS and SRS** are the most important documents to develop any project or software. These types of the document contain the in-depth details of the particular software.

Requirement vs Specification

Requirement	Specification
They plan the software from the end-user, business, and stakeholder perspectives.	They prepare the software from the technical team's point of view.
The requirements define what the software must do.	The specification defines how the software will be developed.
Some of the common terminologies used for requirement document are as follows: ○ SRD: **System Requirement Document**	Some of the common terminologies used for specification document are as follows: ○ FRS: **Functional Requirement Specifications** ○ SRS: **System Requirement Specifications** ○ CRS: **Configurations Requirements**

○ BRD: **Business Requirement Document**	**Specification** ○ PRS: **Performance Requirements Specifications** ○ RRS: **Reliability Requirements Specifications** ○ CRS: **Compatibility Requirements Specifications**

What is BRS?

The BRS document stands for **Business Requirement Specification**. To create the BRS document, the **Business analyst** will interrelate with the customers. The BRS document includes the business rules, the project's scope, and in-detail client's requirements.

In this type of document, the client describes how their business works or the software they need.

For the CRS, the details will be written in the simple business (English) language by the BA (business analyst), which developers and the test engineers cannot understand.

What is SRS?

The SRS document stands for **Software Requirement Specification**.

In this document, the Business Analyst will collect the Customer Requirement Specifications (CRS) from the client and translate them into Software Requirement Specification (SRS).

The SRS contains how the software should be developed and given by the Business Analyst (BA).

Difference between BRS vs SRS

S.NO.	BRS (Business Requirement Specification)	SRS (Software Requirement Specification)
1.	It is a document that describes the requirements of the client using the non-technical expression.	It determines the specifications of a software product more formally.
2.	It prepares the report of the user connections.	It specifies how the clients communicate with the system with the help of use cases.
3.	In the BRS document, it is not important to contain the references of figures and tables.	It always includes references to illustrations and tables.
4.	The BRS document is obtained from relating to the client's requirements and taking responsibility for them.	The Software Requirement Specification obtain from the Business Requirement specification.
5.	The BRS document involved the product's future scope, keeping in mind the organization's strategies for development plans.	The SRS document does not involve the scope of the product.

Session 10

SOFTWARE TEST METRICS

Software Testing Metrics are the quantitative measures used to estimate the progress, quality, productivity and health of the software testing process. The goal of software testing metrics is to improve the efficiency and effectiveness in the software testing process and to help make better decisions for further testing process .

Types of Test Metric

- **Process Metrics:** It can be used to improve the process efficiency of the SDLC (Software Development Life Cycle).
- **Product Metrics:** It deals with the quality of the software product.
- **Project Metrics:** It can be used to measure the efficiency of a project team or any testing tools being used by the team members.

Test Metrics Life Cycle

Analysis

- Identification of the Metrics
- Define the identified QA Metrics

Communicate

- Explain the need for metric to stakeholder and testing team
- Educate the testing team about the data points to need to be captured for processing the metric

Evaluation

- Capture and verify the data
- Calculating the metrics value using the data captured

Report

- Develop the report with an effective conclusion
- Distribute the report to the stakeholder and respective representative

● Take feedback from stakeholder

How do you know a metric is good?

- ✓ Reliability -refers to the consistency of measurement.
- ✓ Validity -The degree to which a measure actually measure what it was intended to measure.
- ✓ Ease of use -How easy it is to capture and use the measurement data.
- ✓ Timelines -refers to whether the data was reported in sufficient time to impact the decision to manage effectively.

Session 11

TYPES OF TESTING

Software Testing Type is a classification of different testing activities into categories, each having a defined test objective, test strategy, and test deliverable. The goal of having a testing type is to validate the Application Under Test(AUT) for the defined Test Objective.

1. Acceptance Testing

Formal testing conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system. It is usually performed by the customer.

2. Ad-hoc Testing

Testing performed without planning and documentation - the tester tries to 'break' the system by randomly trying the system's functionality. It is performed by the testing team.

3. Alpha Testing

Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users.

4. Beta Testing

Final testing before releasing an application for commercial purpose.

5. Benefit Realization Testing

Check whether customer requirements are satisfied or not.

6. **Compatibility Testing**

Testing technique that validates how well a software performs in a particular hardware/software/operating system/network environment. It is performed by the testing teams.

7. **Comparison Testing**

Testing technique which compares the product strengths and weaknesses with previous versions or other similar products. Can be performed by testers, developers, product managers or product owners.

8. Configuration Testing

Testing technique which determines minimal and optimal configuration of hardware and software, and the effect of adding or modifying resources such as memory, disk drives and CPU.

9. Endurance Testing

Type of testing which checks for memory leaks or other problems that may occur with prolonged execution. It is usually performed by performance engineers.

10. Exhaustive Testing

Process of testing the application with all possible inputs.

11. Exploratory Testing

Black box testing technique performed without planning and documentation. It is usually performed by manual testers.

12. Functional Testing

Type of black box testing that bases its test cases on the specifications of the software component under test. It is performed by testing teams.

13. GUI software Testing

The process of testing a product that uses a graphical user interface, to ensure it meets its written specifications. This is normally done by the testing teams.

14. Install/uninstall Testing

Quality assurance work that focuses on what customers will need to do to install and set up the new software successfully. It may involve full, partial or upgrades install/uninstall processes and is typically done by the software testing engineer in conjunction with the configuration manager.

15. Integration Testing

The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

16. Load Testing

Testing technique that puts demand on a system or device and measures its response. It is usually conducted by the performance engineers.

17. Non-functional Testing

Testing technique which focuses on testing of a software application for its non-functional requirements. Can be conducted by the performance engineers or by manual testing teams.

18. Negative Testing

Also known as "test to fail" - testing method where the tests' aim is showing that a component or system does not work. It is performed by manual or automation testers.

19. Performance Testing

Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements.

20. Positive Testing

Testing is done using the valid inputs

21. Ramp Testing

Type of testing consisting in raising an input signal continuously until the system breaks down. It may be conducted by the testing team or the performance engineer.

22. Regression Testing

Type of software testing that seeks to uncover software errors after changes to the program (e.g. bug fixes or new functionality) have been made, by retesting the program. It is performed by the testing teams.

23. Recovery Testing

Testing technique which evaluates how well a system recovers from crashes, hardware failures, or other catastrophic problems. It is performed by the testing teams.

24. Retesting

After rectification the tester again tests the application .

25. Security Testing

A process to determine that an information system protects data and maintains functionality as intended. It can be performed by testing teams or by specialized security-testing companies.

26. Sanity Testing

Testing technique which determines if a new software version is performing well enough to accept it for a major testing effort. It is performed by the testing teams.

27. Smoke Testing

Testing technique which examines all the basic components of a software system to ensure that they work properly. Typically, smoke testing is conducted by the testing team, immediately after a software build is made.

28. Stress Testing

Testing technique which evaluates a system or component at or beyond the limits of its specified requirements. It is usually conducted by the performance engineer.

29. System Testing

The process of testing an integrated hardware and software system to verify that the system meets its specified requirements. It is conducted by the testing teams in both development and target environments.

30. Thread Testing

A variation of top-down testing technique where the progressive integration of components follows the implementation of subsets of the requirements. It is usually performed by the testing teams.

31. User Interface Testing

Type of testing which is performed to check how userfriendly the application is. It is performed by testing teams.

32.Usability Testing

Testing technique which verifies the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component. It is usually performed by end users.

33.Use case Testing

Test can be derived from use case.Use case describes interaction between user and system which produce a result value to a system user or customer.

34.Vendor validation testing

This can be conducted jointly by software vendor and testing team.To ensure that all the requirement functionalities have been delivered.

35.Vulnerability Testing

Type of testing which regards application security and has the purpose to prevent problems which may affect the application integrity and stability. It can be performed by the internal testing teams or outsourced to specialized companies.

Session 12

MOBILE APPLICATION TESTING

Types of Mobile Testing

There are broadly 2 kinds of testing that take place on mobile devices:

1. Hardware testing:

The device including the internal processors, internal hardware, screen sizes, resolution, space or memory, camera, radio, Bluetooth, WIFI etc. This is sometimes referred to as, simple “Mobile Testing”.

2. Software or Application testing:

The applications that work on mobile devices and their functionality are tested. It is called the “Mobile Application Testing” to differentiate it from the earlier method. Even in mobile applications, there are few basic differences that are important to understanding:

a) Native apps: A native application is created for use on a platform like mobile and tablets.

b) Mobile web apps are server-side apps to access website/s on mobile using different browsers like Chrome, Firefox by connecting to a mobile network or wireless network like WIFI.

c) Hybrid apps are combinations of native app and web app. They run on devices or offline and are written using web technologies like HTML5 and CSS.

There are few basic differences that set these apart:

● Native apps have single platform affinity while mobile web apps have cross platform affinity.

● Native apps are written in platforms like SDKs while Mobile web apps are written with web technologies like HTML, CSS, asp.net, Java, PHP.

● For a native app, installation is required but for mobile web apps, no installation is required.

● A native app can be updated from the play store or app store while mobile web apps are centralized updates.
● Many native apps don't require an Internet connection but for mobile web apps, it's a must.
● Native apps work faster when compared to mobile web apps.
● Native apps are installed from app stores like Google play store or app store where mobile web are websites and are only accessible through the Internet.

The significance of Mobile Application Testing

Testing applications on mobile devices is more challenging than testing web apps on the desktop due to

● **Different range of mobile devices** with different screen sizes and hardware configurations like a hard keypad, virtual keypad (touch screen) and trackball etc.
● **Wide varieties of mobile devices** like HTC, Samsung, Apple and Nokia.
● **Different mobile operating systems** like Android, Symbian, Windows, Blackberry and IOS.
● **Different versions of operation system** like iOS 5.x, iOS 6.x, BB5.x, BB6.x etc.
● **Different mobile network operators** like GSM and CDMA.
● Frequent updates – (like Android- 4.2, 4.3, 4.4, iOS-5.x, 6.x) – with each update a new testing cycle is recommended to make sure no application functionality is impacted.

As with any application, Mobile application testing is also very important, as the client is usually in millions for a certain product – and a product with bugs is never appreciated. It often results in monetary losses, legal issues, and irreparable brand image damage.

Basic Difference Between Mobile and Desktop Application Testing:

Few obvious aspects that set mobile app testing apart from the desktop testing

- On the desktop, the application is tested on a central processing unit.
- On a mobile device, the application is tested on handsets like Samsung, Nokia, Apple, and HTC.

● Mobile device screen size is smaller than a desktop.

● Mobile devices have less memory than a desktop.

● Mobiles use network connections like 2G, 3G, 4G or WIFI where desktop use broadband or dial-up connections.

● The automation tool used for desktop application testing might not work on mobile applications.

Types of Mobile App Testing:

● **Usability testing**– To make sure that the mobile app is easy to use and provides a satisfactory user experience to the customers

● **Compatibility testing**– Testing of the application in different mobiles devices, browsers, screen sizes and OS versions according to the requirements.

● **Interface testing**– Testing of menu options, buttons, bookmarks, history, settings, and navigation flow of the application.

● **Services testing**– Testing the services of the application online and offline.

● **Low-level resource testing**: Testing of memory usage, auto-deletion of temporary files, local database growing issues known as low-level resource testing.

● **Performance testing**– Testing the performance of the application by changing the connection from 2G, 3G to WIFI, sharing the documents, battery consumption, etc.

● **Operational testing**– Testing of backups and recovery plan if a battery goes down, or data loss while upgrading the application from a store.

● **Installation tests–** Validation of the application by installing /uninstalling it on the devices.

● **Security Testing**– Testing an application to validate if the information system protects data or not.

Session 13

WEB APPLICATION TESTING

WEB TESTING, or website testing is checking your web application or website for potential bugs before its made live and is accessible to the general public. Web Testing checks for functionality, usability, security, compatibility, performance of the web application or website.

1. Functionality Testing of a Website

Functionality Testing of a Website is a process that includes several testing parameters like user interface, API, database testing, security testing, client and server testing and basic website functionalities. Functional testing is very convenient and it allows users to perform both manual and automated testing. It is performed to test the functionalities of each feature on the website.

Web based Testing Activities includes:

Test all **links** in your web pages are working correctly and make sure there are no broken links. Links to be checked will include -

● Outgoing links

● Internal links

● Anchor Links

● Mail To Links

Test Forms are working as expected. This will include-

● Scripting checks on the form are working as expected. For example- if a user does not fill a mandatory field in a form an error message is shown.

● Check default values are being populated

● Once submitted, the data in the forms is submitted to a live database or is linked to a working email address

● Forms are optimally formatted for better readability

Test Cookies are working as expected. Cookies are small files used by websites to primarily remember active user sessions so you do not need to log in every time you visit a website.

Cookie Testing will include

● Testing cookies (sessions) are deleted either when cache is cleared or when they reach their expiry.

● Delete cookies (sessions) and test that login credentials are asked for when you next visit the site.

Test HTML and CSS to ensure that search engines can crawl your site easily. This will include

● Checking for Syntax Errors

● Readable Color Schemas

● Standard Compliance. Ensure standards such W3C, OASIS, IETF, ISO, ECMA, or WS-I are followed.

Test business workflow- This will include

● Testing your end - to - end workflow/ business scenarios which takes the user through a series of web pages to complete.

● Test negative scenarios as well, such that when a user executes an unexpected step, appropriate error message or help is shown in your web application.

Tools that can be used: **QTP** , IBM Rational , **Selenium**

2. Usability testing:

Usability Testing has now become a vital part of any web based project. It can be **carried out by testers** like you **or a small focus group** similar to the target audience of the web application.

Test the site **Navigation**:

● Menus, buttons or Links to different pages on your site should be easily visible and consistent on all web pages.

Test the **Content**:

● Content should be legible with no spelling or grammatical errors.

● Images if present should contain an "alt" text

Tools that can be used: Chalkmark, Clicktale, Clixpy and Feedback Army

3.Interface Testing:

Three areas to be tested here are - Application, Web and Database Server

● **Application:** Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the application and must be only shown to the administrator and not the end user.

● **Web Server**: Test Web server is handling all application requests without any service

denial.

● **Database Server:** Make sure queries sent to the database give expected results.

Test system response when **connection between the three layers** (Application, Web and Database) **cannot be established** and appropriate message is shown to the end user.

Tools that can be used: AlertFox, Ranorex

4. Database Testing:

Database is one critical component of your web application and stress must be laid to test it thoroughly. Testing activities will include-

● Test if any errors are shown while executing queries

● Data Integrity is maintained while creating, updating or deleting data in the database.

● Check response time of queries and fine tune them if necessary.

● Test data retrieved from your database is shown accurately in your web application

Tools that can be used: **QTP**, **Selenium**

5. Compatibility testing.

Compatibility tests ensure that your web application displays correctly across different devices. This would include

Browser Compatibility Test: Same website in different browsers will display differently.

You need to test if your web application is being displayed correctly across browsers, JavaScript, AJAX and authentication is working fine. You may also check for Mobile Browser Compatibility.

The rendering of web elements like buttons, text fields etc. changes with change in **Operating System**. Make sure your website works fine for various combination of Operating systems such as Windows, Linux, Mac and Browsers such as Firefox, Internet Explorer, Safari etc.

Tools that can be used: Net Mechanic

6. Performance Testing:

This will ensure your site works under all loads. Software Testing activities will include but not limited to -

● Website application response times at different connection speeds

● Load test your web application to determine its behavior under normal and peak loads

● Stress test your web site to determine its break point when pushed to beyond normal loads at peak time.

● Test if a crash occurs due to peak load, how does the site recover from such an event

● Make sure optimization techniques like zip compression, browser and server side cache enabled to reduce load times

Tools that can be used: **Loadrunner**, **JMeter**

7. Security testing:

Security Testing is vital for e-commerce websites that store sensitive customer information like credit cards. Testing Activities will include-

● Test unauthorized access to secure pages should not be permitted

● Restricted files should not be downloadable without appropriate access

● Check sessions are automatically killed after prolonged user inactivity

● On use of SSL certificates, websites should redirect to encrypted SSL pages.

Session 14,15,16

STLC [SOFTWARE TESTING LIFE CYCLE]

1.Requirement Analysis

Study requirement from customer

2.Test Planning

Prepare plan for test entire road map of testing.The document created in test planning is called test plan.What should be covered,not covered are stored in test plan.

a) Test Scope :what to be tested and what not to be tested.

b) Test objective : Find maximum number of defects and also know the purpose of the application.

c) Assumption :when to start/stop it

d) Risk Analysis :Analysis risk during test how to handle it.

e) Test Design : Which level,technique,type are used.

3.Test Case Preparation

Set of procedures executed in a system to identify defects.

Test case id	Test case description	Test case procedure	External input	Expected result	Actual result	Status

4.Test case execution

Evaluate test case preparation table.

5.Test Log

Test log is used to identify the status of the test cases.Pass/Fail information will be stored in this test log.

Test case id	Test case description	Status

6.Defect Tracking/Bug Report :

All the failed items will come under the defect tracking

Defect id	Test case id	Defect description actual result in test case preparation	Defect status	Defect severity	Defect priority	Screenshot/link	Inspected by TL	Inspected By TE

Severity

a) Blocker :-System hang or crash issues

b) critical/major :- data loss,problems related to security,intended function do not work issues

c) Minor :-we can do functions but its alternate method does not work (eg: keyboard shutdown is not working).

d) Trivial :- cosmetic errors like spelling mistakes,alignment is not proper.

e) Enhancement :-modify or can do updation.

Priority

a) High Priority -blocker

b) Critical - crashes,loss of data, several memory leak

c) Medium Priority- critical/major,minor

d) Low Priority -trivial,enhancement

Defect Reporting Template

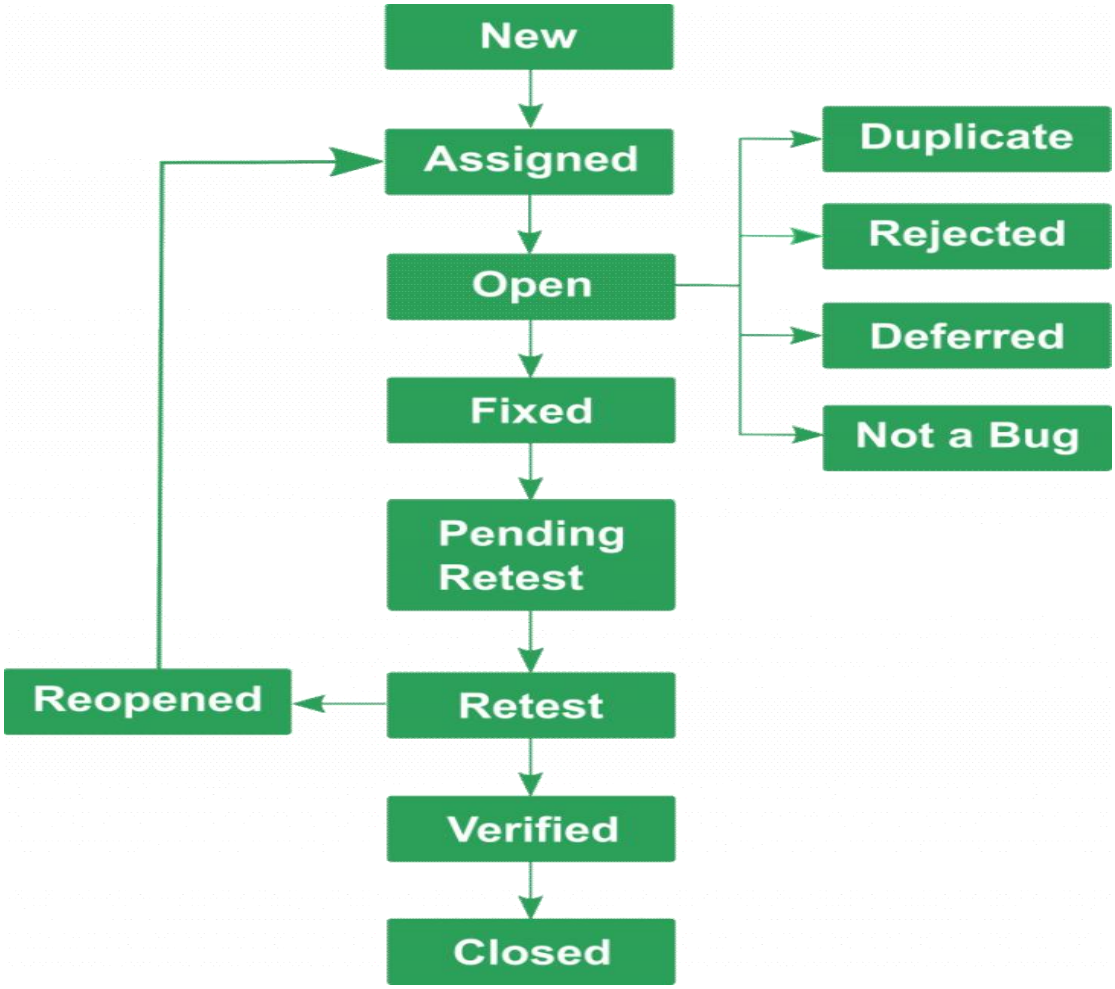
- **Defect Id** – A unique identifier of the defect.
- **Summary** – A one-line summary of the defect, more like a defect title.
- **Description** – A detailed description of the defect.
- **Build Version** – Version of the build or release in which defect is found.
- **Steps to reproduce** – The steps to reproduce the defect.
- **Expected Behavior** – The expected behavior from which the application is deviating because of the defect.
- **Actual Behavior** – The current erroneous state of the application w.r.t. the defect.

- **Priority** – Based on the urgency of the defect, this field can be set on a scale of P0 to P3.
- **Severity** – Based on the criticality of the defect, this field can be set to minor, medium, major or show stopper.
- **Reported By** – Name of the QA, reporting the defect.
- **Reported On** – The date on which the defect was raised.
- **Assigned To** – The person to whom the defect is assigned in its current state. It can be the developer fixing the defect, the QA for verification of the fixed defect or the manager approving the defect.
- **Current Status** – The current status of the defect (one of the states of the defect life cycle).
- **Environment** – The environment in which the defect is found – release, staging, production, etc.

Bug Life cycle

The bug report life cycle begins with the bug being detected and reported by the tester and ends after closure. Over the entire life cycle of the bug has different states.

The schematic **life cycle** can be shown on this **graphic**:



New: When a bug is reported and posted for the first time. Its state is given as new.

1. **Assigned:** After the tester has reported the bug, the lead of the tester confirms that the defect is valid and it is assigned to the appropriate developer or developers team.

2. **Open:** It means that the developer has begun to analyze the bug and try to fix it.

3. **Fixed:** After a developer changes the code and fixes a bug, they change state to "Fixed" and it can be passed to the QA team for retesting.

4. **Pending Retest:** At this stage bug report waiting for retesting.

5. **Retest:** At this stage, the testers check the amendments and retest the changes that developers have made.

6. **Verified:** If retesting it isn't detected the bug and the product is working properly, the tester changes the bug report status to "Verified".

7. **Reopen:** Reopen: In case the tester rechecked and the bug still exists, the state of bug becoming “Reopen” and bug report goes through the life cycle once again.

8. **Closed:** Once the developer has corrected the mistakes, he sends the product to testers

for retesting. If the tester decides that the bug is fixed, he or she changes the bug

report status to “Closed.” This means that the defect is fixed, checked and approved.

9. **Duplicate:** If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to “duplicate**“.**

10.**Rejected:** If the developer feels that the bug is not genuine, he rejects the bug. Then the state of the bug is changed to “rejected”.

11.**Deferred:** this means that the bug will be fixed, but in another release, and now it’s waiting. Usually, the reason for this is the low priority of bugs and lack of time.

12.**Not a bug:** Bag report can have that status, in the case of, for example, if a customer asked to make any little changes to the product: change colour, font, and more.

7.Test Report

Test Report is a document which contains a summary of all test activities and final test results of a testing project. Test report is an assessment of how well the Testing is performed. Based on the test report, stakeholders can evaluate the quality of the tested product and make a decision on the software release.

For example, if the test report informs that there are many defects remaining in the product, stakeholders can delay the release until all the defects are fixed.Test report is a **communication** tool between the Test

Manager and the stakeholder. Through the test report, the stakeholder can **understand** the project situation, the quality of product and other things.

Eight Interim Report

1. Functional testing status : this report will show percentages of the function which have been

-fully tested

-tested with open defects

-not tested

2. Functions working timeline :this report will show the actual plan to have all functions working verses the current status of the functions working.An ideal format could be a line graph.
3. Expected verses Actual defects detected: this report will provide an analysis between the number of defects being generated against the expected number of defects expected from planning stage.
4. Defect Detected verses Corrected gap :this report,ideally in a line graph format ,will show the number of defects uncovered verses the number of defects being corrected and accepted by the testing group.if the gap grows too large,he project may not be ready when originally planned.
5. Average age uncorrected defects by type : this report will show the average days of outstanding defects by type[sev1,sev2etc].In the planning stage,it is a beneficial to determine the acceptable open days by defect type.
6. Defect Distribution : This report will show the defect distribution by function or module.it can also show items such as numbers of tests completed.
7. Relative €defect Distribution ; this report will take the previous report(defect distribution) and normalize the level of defect.

8. Testing Acton : this report can show many different things including possible shortfalls in testing.

Session 17

TEST CASE

The test case is defined as a group of conditions under which a tester determines whether a software application is working as per the customer's requirements or not. Test case designing includes preconditions, case name, input conditions, and expected result. A test case is a first level action and derived from test scenarios.