

## SOFTWARE TESTING

Software testing is the process of ensuring that a developed software is performing its functionalities as expected. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements. Testing is a process of evaluating a system by manual or automatic and verifying that it satisfies that specified requirements or identifying the difference between expected result and actual result.

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

### **Two ways of Testing**

#### 1. Manual Testing

It is the process of manually testing a software application for defects. It requires a tester to play the role of an end user and use most of all the features of the application to ensure correct behaviour. The main goal of manual testing is to make sure that the application under test is defect free.

#### 2.Automation Testing

It is the process of testing a software application by using a software tool. It is the process in which a tool executes pre-scripted tests on a software application before it is released into production.

**Automation Testing** is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

### Difference Between Manual & Automation Testing

Parameter	Automation Testing	Manual Testing
Definition	Automation Testing uses automation tools to execute test cases.	In manual testing, test cases are executed by a human tester and software.
Processing time	Automated testing is significantly faster than a manual approach.	Manual testing is time-consuming and takes up human resources.
Exploratory Testing	Automation does not allow random testing	Exploratory testing is possible in Manual Testing
Initial investment	The initial investment in the automated testing is higher. Though the ROI is better in the long run.	The initial investment in the Manual testing is comparatively lower. ROI is lower compared to Automation testing in the long run.
Reliability	Automated testing is a reliable method, as it is performed by tools and scripts. There is no testing Fatigue.	Manual testing is not as accurate because of the possibility of the human errors.
UI Change	For even a trivial change in the UI of the AUT, Automated Test Scripts need to be modified to work as expected	Small changes like change in id, class, etc. of a button wouldn't thwart execution of a manual tester.
Investment	Investment is required for testing tools as well as automation engineers	Investment is needed for human resources.

### Why do we need testing?

- Ensure that software is bug free
- Ensure that the system meets customer requirements and software specifications.
- Ensure that the system meets end user expectations.
- Fixing the bugs identified after release, is expensive

### What are the benefits of Software Testing?

- **Cost-Effective**: It is one of the important advantages of software testing. Testing any IT project on time helps you to save your money for the long term. In case if the bugs caught in the earlier stage of software testing, it costs less to fix.
- **Security**: It is the most vulnerable and sensitive benefit of software testing. People are looking for trusted products. It helps in removing risks and problems earlier.
- **Product quality**: It is an essential requirement of any software product. Testing ensures a quality product is delivered to customers.

- Customer Satisfaction: The main aim of any product is to give satisfaction to their customers. UI/UX Testing ensures the best user experience.

### Seven principles of testing

- ★ Testing Shows the presence of defects not their absence : Testing shows the presence of defects in the software. The goal of testing is to make the software fail. Sufficient testing reduces the presence of defects. In case testers are unable to find defects after repeated regression testing doesn't mean that the software is bug-free. Testing talks about the presence of defects and doesn't talk about the absence of defects.
- ★ Exhaustive Testing is Impossible : Testing everything (i.e. all combinations of input and preconditions) is not feasible for trivial cases.
- ★ Testing should start as early as possible.
- ★ Defect ~~Clustering~~ In software testing means that a small module or functionality contains most of the bugs or it has the most operational failures.
- ★ Pesticide Paradox: Pesticide Paradox in software testing is the process of repeating the same test cases again and again, eventually, the same test cases will no longer find new bugs. So to overcome this Pesticide Paradox, it is necessary to review the test cases regularly and add or update them to find more defects.
- ★ Testing is more context dependent: Testing approach depends on the context of the software we develop. We do test the software differently in different contexts. For example, an online banking application requires a different approach to testing compared to an e-commerce site.
- ★ Absence of error-Fallacy : Finding and fixing errors does not help if the system built is unusable and does not fulfil the users needs and expectations.

## Module 1

### Defect

A defect is an undesirable state.

#### *Categories of Defect :*

- Extra - An extra requirement incorporated into the product that was not requested by the client.
- Wrong - Requirements have been implemented in the wrong way. This defect is a variance from the given specification.
- Missing - A requirement which has been requested by the client but not implemented in the application.

### Error

It refers to the difference between actual output and expected output. Or The mismatch between the application and its specification.

#### *Types:*

- *Error*: Mistake made by people ; usually reported by developers.

- *Fault* : It is an incorrect step, process or data definition in a computer program which causes the program to behave in an unintended or unanticipated manner.
- *Bug*: It is a fault in the program which causes the program to behave in an unintended or unanticipated manner reported by engineer(s).
- *Defect* : An undesirable state is an error in coding or logic that causes a program to malfunction or to produce unexpected or incorrect results.
- *Failure* :It is the inability of a system or component to perform a required function, according to its specifications or specified performance requirements. Failure occurs when a fault is executed

### QUALITY

Quality is defined as meeting the customers requirements first time and every time. Quality is much more than the absence of defects which allows us to meet customers expectations.

#### Five perspectives of quality

- ❖ Transcendent : I know it when I see it.
- ❖ Product based: Possesses desired features.
- ❖ User based: Fitness for use.
- ❖ Development & manufacturing based : conforms the requirements
- ❖ Value based: At an acceptable cost.

### SOFTWARE QUALITY FACTORS (SQF)

- **Correctness** : These requirements deal with the correctness of the output of the software system. They include – Output mission The required accuracy of output that can be negatively affected by inaccurate data or inaccurate calculations. The completeness of the output information, which can be affected by incomplete data. The up-to-dateness of the information defined as the time between the event and the response by the software system. The availability of the information. The standards for coding and documenting the software system.

- **Reliability** :Reliability requirements deal with service failure. They determine the maximum allowed failure rate of the software system, and can refer to the entire system or to one or more of its separate functions. Efficiency It deals with the hardware resources needed to perform the different functions of the software system. It includes processing capabilities (given in MHz), its storage capacity (given in MB or GB) and the data communication capability (given in MBPS or GBPS). It also deals with the time between recharging of the system's portable units, such as, information system units located in portable computers, or meteorological units placed outdoors. Integrity This factor deals with the software system security, that is, to prevent access to unauthorized persons, also to distinguish between the group of people to be given read as well as write permit.
- **Usability** : Usability requirements deal with the staff resources needed to train a new employee and to operate the software system.
- **Maintainability** This factor considers the efforts that will be needed by users and maintenance personnel to identify the reasons for software failures, to correct the failures, and to verify the success of the corrections.
- **Flexibility** :This factor deals with the capabilities and efforts required to support adaptive maintenance activities of the software. These include adapting the current software to additional circumstances and customers without changing the software. This factor's requirements also support perfective maintenance activities, such as changes and additions to the software in order to improve its service and to adapt it to changes in the firm's technical or commercial environment.
- **Testability** :Testability requirements deal with the testing of the software system as well as with its operation. It includes predefined intermediate results, log files, and also the automatic diagnostics performed by the software system prior to starting the system, to find out whether all components of the system are in working order and to obtain a report about the detected faults. Another

type of these requirements deals with automatic diagnostic checks applied by the maintenance technicians to detect the causes of software failures.

- **Reusability** : This factor deals with the use of software modules originally designed for one project in a new software project currently being developed. They may also enable future projects to make use of a given module or a group of modules of the currently developed software. The reuse of software is expected to save development resources, shorten the development period, and provide higher quality modules.
- **Interoperability**: Interoperability requirements focus on creating interfaces with other software systems or with other equipment firmware.

### Quality Assurance

QA includes activities that ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements. Focuses on processes and procedures rather than conducting actual testing on the system.

#### Activities

1. Auditing
2. Monitoring
3. Testing

#### Objectives

1.

1. **Goals**: Focusing on goals verifies that the system achieves the objectives of both the user and total organization.
2. **Methods** : Method verification proves that the structured methodologies are adhered to and the policies, procedures, standards and guidelines that are in place are being followed.
3. **Performance** : is monitored to prove that the most optimal use of computer hardware and software is in place when implementing the application. Quality Control It includes activities that ensure the verification of a developed software with respect to documented (or not in

some cases) requirements. Focuses on actual testing by executing the software with an aim to identify bugs/defects through implementation of procedures and processes.

### Difference between QA & QC

Quality Assurance (QA)	Quality Control (QC)
<ul style="list-style-type: none"><li>• It is a procedure that focuses on providing assurance that quality requested will be achieved</li></ul>	<ul style="list-style-type: none"><li>• It is a procedure that focuses on fulfilling the quality requested.</li></ul>
<ul style="list-style-type: none"><li>• QA aims to prevent the defect</li></ul>	<ul style="list-style-type: none"><li>• QC aims to identify and fix defects</li></ul>
<ul style="list-style-type: none"><li>• It is a method to manage the quality- Verification</li></ul>	<ul style="list-style-type: none"><li>• It is a method to verify the quality-Validation</li></ul>
<ul style="list-style-type: none"><li>• It does not involve executing the program</li></ul>	<ul style="list-style-type: none"><li>• It always involves executing a program</li></ul>
<ul style="list-style-type: none"><li>• It's a Preventive technique</li></ul>	<ul style="list-style-type: none"><li>• It's a Corrective technique</li></ul>
<ul style="list-style-type: none"><li>• It's a Proactive measure</li></ul>	<ul style="list-style-type: none"><li>• It's a Reactive measure</li></ul>
<ul style="list-style-type: none"><li>• It is the procedure to create the deliverables</li></ul>	<ul style="list-style-type: none"><li>• It is the procedure to verify that deliverables</li></ul>

### Quality Control

QC testing is a function of software quality that checks that the project follows standards, processes, and procedures laid down, and that the project produces the required internal and external deliverables. The QC team's job is to identify defects after a product is developed but before it is released. QC aims to identify (and correct) defects in the finished product.

### Quality Testing



## Soften Technologies

The IEEE defines testing as a process of operations under specified conditions, observing or recording the result and making an evaluation of some aspect of a small component.

QT=**VERIFICATION+VALIDATION**

### **Verification:**

The process of evaluating a s/m or component or determining whether the product as a given development phase satisfies the conditions imposed and status of the phase.

### **ARE WE BUILD THE PRODUCT RIGHT?**

-Ensure that the software is developed in the right manner.

### Validation:

The process of evaluating a s/m or component during or at the end of the development process to determine whether it satisfies specified requirements

### **ARE WE BUILD THE RIGHT PDT?**

-Ensure you build one right things or product

## **COST OF QUALITY**

All the costs that occur beyond one cost of producing the product right at the first time.

### *1.Prevention cost*

Money required to prevent errors and to do the job right the first time .The category includes money spent on establishing methods and producers, training workers, acquiring tools and planning or quality. Preventing money is all spent before the product is actually built.

### *2.Appraisal cost*

## Soften Technologies

Money spent to review completed products against requirement appraisal includes the cost of inspections, testing and reviews. The money spent after the product is built but before it is shipped to the user.

### 3.Failure cost

This cost is associated with defective products that have been delivered to the user or moved into production. Some failure costs involve repairing products to meet these requirements..

#### 1.Internal Failure Cost

#### 2.External Failure Cost

##### Internal failure cost

Quality cost associated with defects discovered before the product has been delivered to customer or moved into production .These internal costs are detected through the firm's inspection and appraisal activities and may include cost of rework scrapping of rejected products downtime caused by quality problems.

##### Alpha testing

Customer tests the application at developers site with/without the presence of a tester .It implies an initial meeting between software vendor and client to ensure that client's requirements are properly met by the developer in terms of performance ,functionality and durability of the software program.

##### Beta testing

>takes place at customer's site

>also known as yield testing

## Soften Technologies

>its goal is to place the application in the hands of real users to discover any flows and issues from the user's perspective that would not want to have in the final released version of the application

External failure cost

External failure cost arise from the rejection of products or services by the customers due to poor quality of the product shipped .liability from legal actions /penalties

repairs and replacement

warranty or work

product recall

loss of market share

loss of business

## MODELS

- Waterfall model
- Spiral model
- Verification and validation model
- Iterative model
- Incremental model
- Agile Model

## Agile Methodology

## Soften Technologies

Agile sdlc method is a combination of iterative and incremental process model with focus on process adaptability and customer satisfaction by rapid delivery of working software product agile methods breaks the product into small incremental builds .Every interaction involves cross functional teams working simultaneously or various area like ,

1.planning

2.requirement analysis

3.design

4.coding

5.unit testing

6.acceptance testing

### *Agile manifesto principles*

> Individuals and interactions

> working software

> customer collaboration

> responding to change

### *Advantages*

> Customer satisfaction by rapid ,continuous delivery of useful software

> Face to face conversation is best form of communication

> Continues attention to technical excellence is good design

> Regular adaptation to changing circumstances

> Even late changes in requirements are welcomed

### ***12 principles agile methodology***

1. Welcome change
2. Satisfy the customer
3. Deliver frequently
4. Work together
5. Build projects
6. Face to face time
7. Measure of progress
8. Sustainable development
9. Continuous attention
10. Keep it simple
11. Organised team
12. Reflect for effectiveness

### **Iterative Model**

The iterative process model is the implementation of the software development life cycle in which the initial development is started based on the initial requirements and more features are added to the base software product with the ongoing iterations until the final system is created.

### Features of Iterative Model

The iterative model was designed as an improvement to the existing waterfall model. The waterfall model is a linear SDLC model whereas the iterative model is cyclical in nature.

Once the initial requirement planning process is completed, some of the other stages are repeated. As these cycles are completed and implemented, the overall end product is improved and iterated on.



- The first stage is the planning stage. It is used to map out the particular requirements. Be it either hardware or software. Here, we also prepare for the other stages to follow.
- The second stage is the analysis stage. It is performed to check if the required models, business logic are incorporated into the project or not.
- Then comes the design stage. In this stage, the team on the project should have a complete set of requirements to work from along with the direction which is to be taken for the project and a Conceptual Systems Design.
- The fourth stage is the implementation and coding stage. All the requirements, planning, and design plans are implemented and coded in this stage. This is the point in the project when the actual construction of the system starts. This is the time to start writing the program code for the project.
- The fifth stage is the testing stage. Here the current build iteration is tested against some standards and norms to check if they satisfy them. These testing procedures are set in place to find out any bugs or errors in our system.

There are various types of testing techniques that can be implemented by the team so as to test their system. This includes – performance testing, stress testing, security testing, requirements testing, usability testing, multi-site testing, etc.

- Finally, when all these stages are completed. A meticulous evaluation is done on the system developed up until this stage. The development team and the stakeholders are able to examine the system and give their feedback regarding various aspects of the system.

These stages are repeated if any new requirements pop up, or any error/ bug is identified in our system.

### Advantages of Iterative Model

- This model produces working software much quickly and early during the SDLC.
- This model is very flexible. As new functionality can be added to it at any time of development.
- This model is considerably cheap as it is less costly to change requirements as compared to the other process models.
- The end-user or the stakeholders can give their feedback quickly, which can then be implemented into the system.
- The errors and bugs in the system can be identified early.
- Takes smaller development teams as compared to other process models.

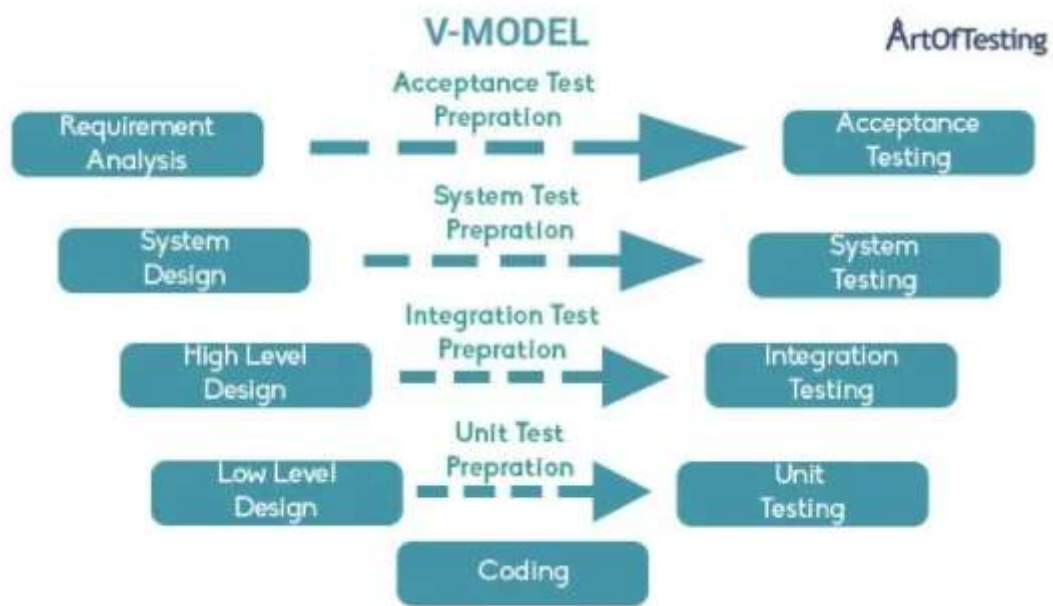
### Disadvantages of Iterative Model

- Problems pertaining to the system architecture can come up because all the requirements are not gathered upfront.
- It is not a good choice for small projects.
- More resource-intensive than waterfall model.
- Risk analysis requires highly qualified specialists to check the risks in our system.
- The whole process is difficult to manage

Once these stages are finished. In other words, all the requirements are fulfilled and meticulously checked. The most recently built iteration of the system is given to the end-user.

## V Model

The V-model of SDLC carries out its execution in a sequential manner. The structure it follows takes the shape of the letter V. This model is also popularly termed as a Verification and Validation model. Here, each phase has to be finished before beginning the next phase. A sequential design progression is followed like that of the waterfall model.



As we can see in the above diagram, the test activities start in parallel with the development activities e.g. during the requirement analysis phase, acceptance-test cases are prepared by the testing team; during the system design phase, the system test case is prepared. Similarly, for each phase of development, a corresponding QA activity is performed. Later, when the deliverable gets ready, the QA artifacts are used to conduct the testing. Along with that, each phase of the development phase is verified before moving to the next phase.

### Advantages of V Model

- Each phase of development is tested before moving to the next phase, hence there is a higher rate of success.
- It avoids defect leakage to the later phases as each phase is verified explicitly.
- The model has clear and defined steps. So, it is easier to implement.



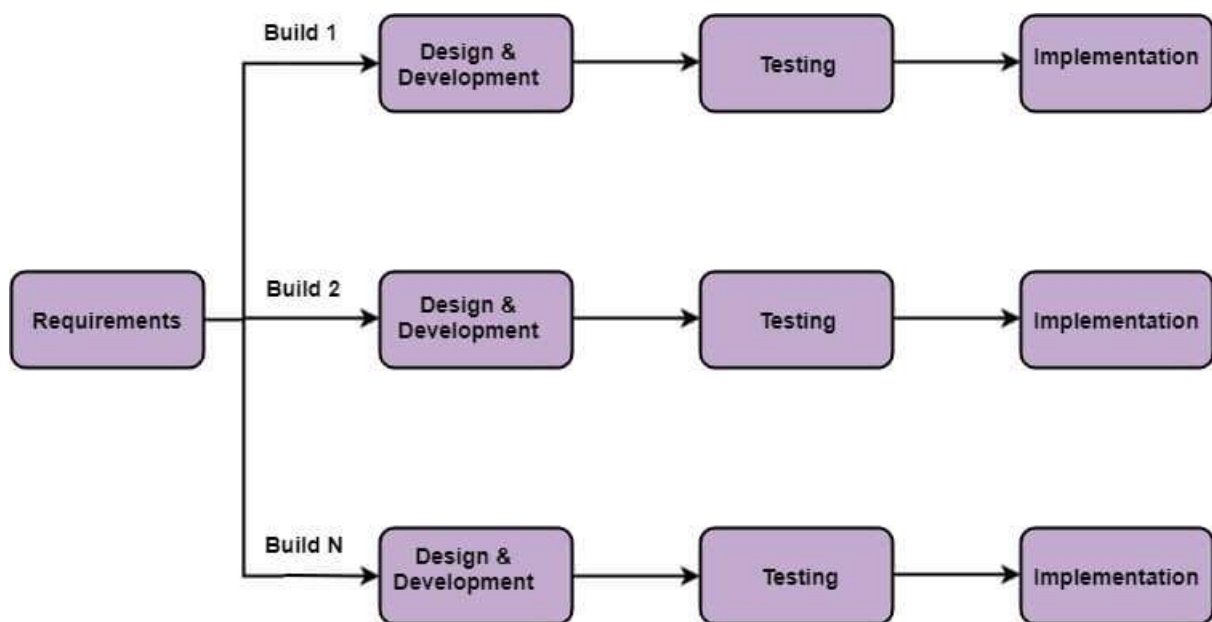
- It is suitable for smaller projects where requirements are fixed.

### Disadvantages of V Model

- The testing team starts in parallel with development. Hence, the overall budget and resource usage increases.
- Changes in requirements are difficult to incorporate.
- The working model of the software is only available in the later phases of the development.
- It is not suitable for complex and large applications because of its rigid process

### **Incremental Model**

Incremental Model is a process of software development where requirements are divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system is achieved.



**Fig: Incremental Model**

The various phases of incremental model are as follows:

**1. Requirement analysis:** In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood

by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.

**2. Design & Development:** In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with success. When software develops new practicality, the incremental model uses style and development phase.

**3. Testing:** In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behaviour of each task.

**4.Implementation:** Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

When we use the Incremental Model?

- When the requirements are superior.
- A project has a lengthy development schedule.
- When Software teams are not very well skilled or trained.
- When the customer demands a quick release of the product.
- You can develop prioritized requirements first.

### Advantage of Incremental Model

- Errors are easy to recognize.
- Easier to test and debug
- More flexible.
- Simple to manage risk because it is handled during its iteration.
- The Client gets important functionality early.

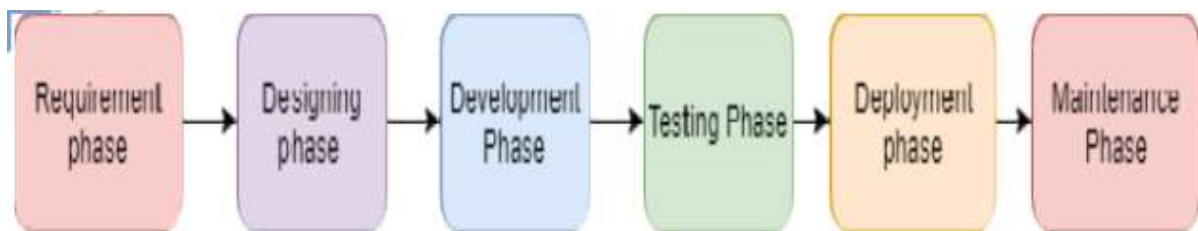
### Disadvantage of Incremental Model

## Soften Technologies

- Need for good planning
- Total Cost is high.
- Well defined module interfaces are needed.

### **SDLC [Software Development Life Cycle]**

SDLC is a process that creates a structure of software development. There are different phases within SDLC, and each phase has its various activities. It makes the development team able to design, create, and deliver a high-quality product. SDLC describes various phases of software development and the order of execution of phases. Each phase requires deliverables from the previous phase in a life cycle of software development. Requirements are translated into design, design into development and development into testing; after testing, it is given to the client.



#### 1. Requirement Phase

This is the most crucial phase of the software development life cycle for the developing team as well as for the project states requirements, specifications, expectations, and any other special requirement related to the product or software. All these are gathered by the business manager or project manager or analyst of the service pro The requirement includes how the product will be used and who will use the product to determine the load of operations. All information gathered from this phase is critical to developing the product as per the customer requirements.

#### 2. Design Phase

The design phase includes a detailed analysis of new software according to the requirement phase. This is the high priority phase in the development life cycle of a system because the

logical designing of the system is converted into physic designing. The output of the requirement phase is a collection of things that are required, and the design phase gives the way to accomplish these requirements. The decision of all required essential tools such as .NET, PHP, a database like Oracle, MySQL, a combination of hardware . This is the most crucial phase of the software development life cycle for the developing team as well as for the project manager. During this phase, the client states requirements, specifications, expectations, and any other special requirement related to the product or software. All these are gathered by the business manager or project manager or analyst of the service providing company. The requirement includes how the product will be used and who will use the product to determine the load of operations. All information gathered from this phase is critical to developing the product as per the customer requirements. The design phase includes a detailed analysis of new software according to the requirement phase. This is the high priority phase in the development life cycle of a system because the logical designing of the system is converted into physic designing. The output of the requirement phase is a collection of things that are required, and the design phase gives the way to accomplish these requirements. techniques and tools, such as data flow diagrams, flowcharts, decision tables, and decision trees, Data dictionary, and the structured dictionary are used for describing the system design.

### 3. Build /Development Phase

After the successful completion of the requirement and design phase, the next step is to implement the design into the development of a software system. In this phase, work is divided into small units, and coding starts by the team of developers according to the design discussed in the previous phase and according to the requirements of the client discussed in the requirement phase to produce the desired result. Front-end developers develop easy and attractive GUI and necessary interfaces to interact with back-end operations and back-end developers do back-end coding according to the required operations. All is done according to the procedure and guidelines demonstrated by the project manager. Since this is the coding phase, it takes the longest time and more focused approach for the developer in the software development life cycle.

### 4. Testing Phase

## Soften Technologies

Testing is the last step of completing a software system. In this phase, after getting the developed GUI and back-end combination, it is tested against the requirements stated in the requirement phase. Testing determines whether the software is actually giving the result as per the requirements addressed in the requirement phase or not.

### 5. Deployment/ Deliver Phase

When software testing is completed with a satisfying result, and there are no remaining issues in the working of the software, it is delivered to the customer for their use. As soon as customers receive the product, they are recommended first to do the beta testing. In beta testing, customers can require any changes which are not present in the software but mentioned in the requirement document or any other GUI changes to make it more user-friendly. Besides this, if any type of defect is encountered while a customer is using the software; it will be informed to the development team of that particular software to sort out the problem. If it is a severe issue, then the development team solves it in a short time; otherwise, if it is less severe, then it will wait for the next version. After the solution of all types of bugs and changes, the software finally deployed to the end-user.

### 6. Maintenance

The maintenance phase is the last and long-lasting phase of SDLC because it is the process which continues until the software's life cycle comes to an end. When a customer starts using software, then actual problems start to occur, and at that time there's a need to solve these problems. This phase also includes making changes in hardware and software to maintain its operational effectiveness like to improve performance, enhance security features and according to customer's requirements with upcoming time. This process to take care of the product from time to time is called maintenance.

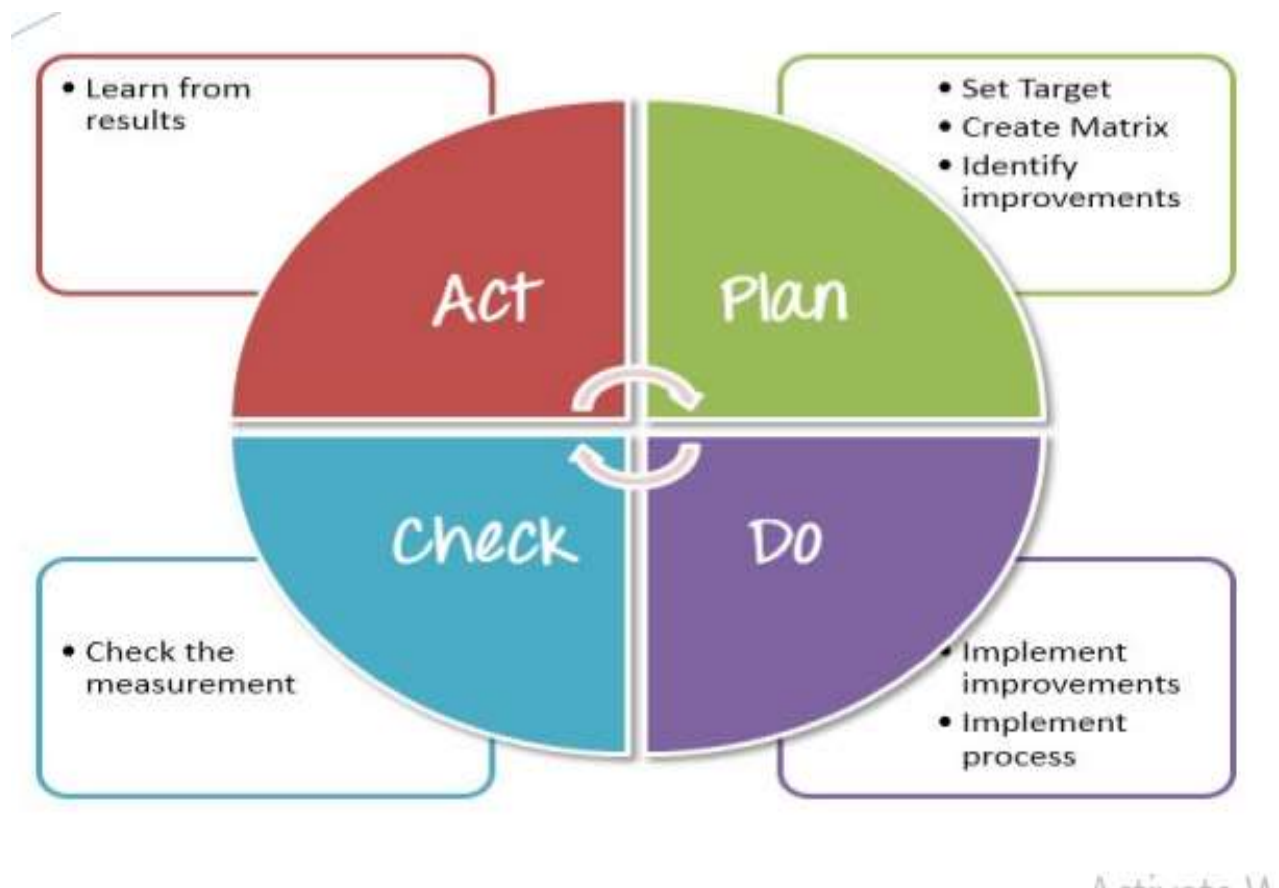
## PROCESS

A process can be defined as a set of activities that represents the way work is performed. The outcome from a process is usually a product or service. These are two ways to visually portray a process:

a) PDCA Cycle

b) Workbench

### *PDCA Cycle*



PLAN :- Define goal and plan for achieving goal.

DO :- Depending on the plan strategy decided during the plan stage we do execution according to this phase.

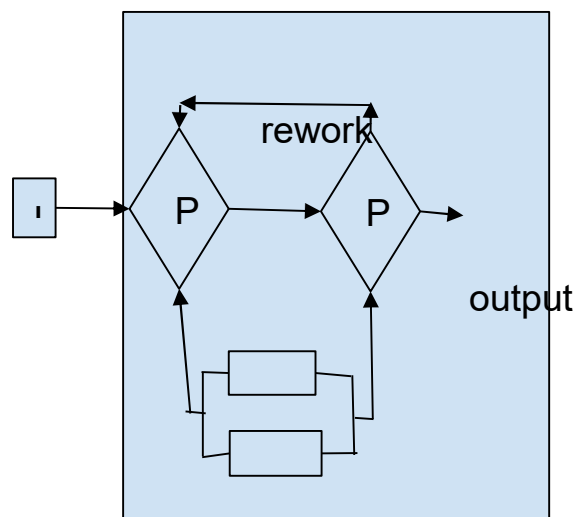
**CHECK :-** Check or test to ensure that we are moving according to plan and are getting the desired result.

**ACT :-** During the check cycle , if any error/issue occurs then we take appropriate action accordingly and revise our plan again.

### ***Workbench***

It is a more practical illustration of a process. The testers workbench is a pictorial representation of how a specific test task is performed. Workbench is built on the following components.

1. Objective –Purpose of the process
2. People skills-Roles, responsibilities and associated skills needed to execute a process.
3. Input –Entrance criteria or deliverable needed to perform testing.
4. Procedure –Do and check procedure.



## **Certifications**

CMMI (Capacity Maturity Model integration)

CMM is a method to evaluate and measure the maturity of the software development process of an organisation.

Certifications are given to organisations by Carnegie Mellon University. for getting this we have to satisfy 18 KPA.

There are 5 maturity levels designated by the number 1 through 5.

CMM 1 -Initial(Lowest quality/Highest risk)

-Any newly satisfied software organisation will get this certification.

CMM 2-Repeatable(Lowest quality/high risk)

-if any organisation managing the existing process or following a new process to develop a software or organisation following trial and error method to develop a software will get the second level certification.

CMM 3-Defined(Medium quality/medium risk)

-if any organisation having some standards and procedures to develop a software will get the third level of certifications.

CMM 4-managing(Higher quality/Lowest risk)

-If any organisation having some target and policy to develop a software will get the fourth level of certifications.



## Soften Technologies

### CMM 5- Optimized

-if any organisation develops a system by reducing the resources and developing the software using new tools and techniques and achieves more will get the fifth level of certifications.

ISO- International Organisation for standard

PCMM- People capability maturity model

-it is given to a particular department of an organisation based on their performance and progress.

TMM-testing maturity model

-it is given to the testing department of an organisation.

5 levels of TMM

1.Initialisation- to make sure that the software is running fine & no defect.

2.Definition- Test strategy,test plan,test cases are developed according to the requirement.

3.Integration- Testing is integrated with software life cycle & becomes a part of it.

4.Management & measurement-Test becomes the part of all activities in the software life cycle.

5.Optimization-carried out by the help of different tools

Six Sigma Certifications-Zero Defect Orientation

### **Test Scenario**

A TEST SCENARIO is defined as any functionality that can be tested. It is also called *Test Condition* or *Test Possibility*. As a tester, you should put yourself in the end user's shoes and figure out the real-world scenarios and use cases of the Application Under Test.

### How to Write Test Scenarios

As a tester, you can follow these five steps to create Test Scenarios-

- **Step 1:** Read the Requirement Documents like BRS, SRS, FRS, of the System Under Test (SUT). You could also refer to use cases, books, manuals, etc. of the application to be tested.
- **Step 2:** For each requirement, figure out possible users actions and objectives. Determine the technical aspects of the requirement. Ascertain possible scenarios of system abuse and evaluate users with hacker's mindset.
- **Step 3:** After reading the Requirements Document and doing your due Analysis, list out different test scenarios that verify each feature of the software.
- **Step 4:** Once you have listed all possible Test Scenarios, a [Traceability Matrix](#) is created to verify that each & every requirement has a corresponding Test Scenario
- **Step 5:** The scenarios created are reviewed by your supervisor. Later, they are also reviewed by other Stakeholders in the project.

### Examples:

Test scenarios of a pen

1. Verify that the length and the diameter of the pen are as per the specifications.
2. Verify the outer body material of the pen. {Check if it is metallic, plastic, or any other material specified in the requirement specifications.}
3. Check the color of the outer body of the pen. It should be as per the specifications.
4. Verify that the brand name and/or logo of the company creating the pen should be clearly visible.
5. Verify that any information displayed on the pen should be legible and clearly visible.

6. Verify the type of pen, whether it is a ballpoint pen, ink pen, or gel pen.
7. Verify that the user is able to write clearly over different types of papers.
8. Check the weight of the pen. It should be as per the specifications. In case not mentioned in the specifications, the weight should not be too heavy to impact its smooth operation.
9. Verify if the pen is with a cap or without a cap.
10. Verify over a surface.
11. Verify the surfaces over which the pen is able to write smoothly color of the ink of the pen.
12. Check the odor of the pen's ink on writing apart from paper e.g. cardboard, rubber surface, etc.
13. Verify that the text written by the pen should have consistent ink flow without leaving any blob.
14. Check that the pen's ink should not leak in case it is tilted upside down.
15. Verify if the pen's ink should not leak at higher altitudes.

Write test scenarios of the following objects:

- .Pencil
- Fan
- Lift
- Door
- Keyboard
- Mouse
- Atm machine
- Google search page
- Gmail login page
- Chair

## Soften Technologies

- Table
- Book
- Wrist watch
- Stapler
- Whatsapp
- Remote control

## **Module 2**

### **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.

## REQUIREMENTS TRACEABILITY MATRIX

	Test Case ID	TC001	TC002	TC003	TC004	TC005	Total Test Case ID 
Requirement ID							
REQ 1		X			X		2
REQ 2		X		X			2
REQ 3		X	X		X	X	4
REQ 4					X		1
REQ 5			X	X	X		3
REQ 6			X			X	2

## Soften Technologies

### 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 & Testcases/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 positive 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.

## 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 deliverables. 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.<sup>2</sup>
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 user-friendly 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. **Usecase Testing:** Test can be derived from usecase. 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.

### Levels of testing

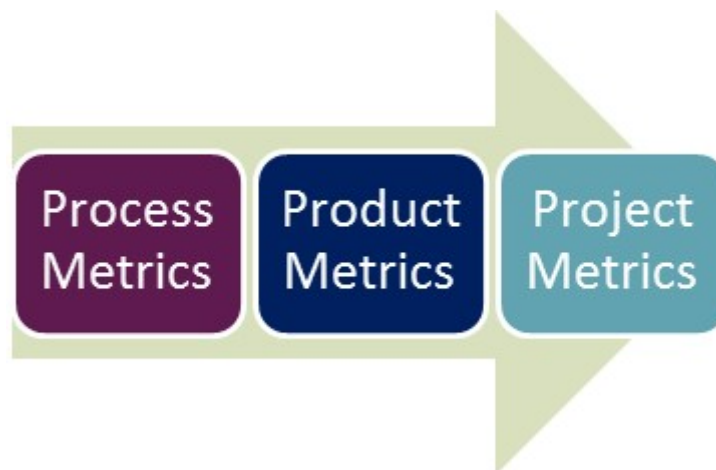
1. **Unit Testing** :-Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.A unit is a single testable part of a software system 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.
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

### 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

Different stages of Metrics life cycle	Steps during each stage
<ul style="list-style-type: none"><li>● Analysis</li></ul>	<ul style="list-style-type: none"><li>● Identification of the Metrics</li><li>● Define the identified QA Metrics</li></ul>
<ul style="list-style-type: none"><li>● Communicate</li></ul>	<ul style="list-style-type: none"><li>● Explain the need for metric to stakeholder and testing team</li><li>● Educate the testing team about the data points to need to be captured for processing the metric</li></ul>
<ul style="list-style-type: none"><li>● Evaluation</li></ul>	<ul style="list-style-type: none"><li>● Capture and verify the data</li><li>● Calculating the metrics value using the data captured</li></ul>

- Report

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

### Difference between Static testing and Dynamic Testing

Static testing	Dynamic testing
In static testing, we will check the code or the application without executing the code.	In dynamic testing, we will check the code/application by executing the code.

Static testing includes activities like code Review, Walkthrough, etc.	Dynamic testing includes activities like functional and non-functional testing such as UT (usability testing), IT (integration testing), ST (System testing) & UAT (user acceptance testing).
Static testing is a <b>Verification</b> Process.	Dynamic testing is a <b>Validation</b> Process.
Static testing is used to prevent defects.	Dynamic testing is used to find and fix the defects.
Static testing is a more cost-effective process.	Dynamic testing is a less cost-effective process.
This type of testing can be performed before the compilation of code.	Dynamic testing can be done only after the executable are prepared.
Under static testing, we can perform the statement coverage testing and structural testing.	Equivalence Partitioning and Boundary Value Analysis techniques are performed under dynamic testing.

It involves the checklist and process which has been followed by the test engineer.	This type of testing required the test case for the execution of the code.
---	--

## 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.

## Dynamic Testing

Dynamic testing is testing, which is done when the code is executed at the runtime environment. It is a validation process 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.

## 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 users and other systems.

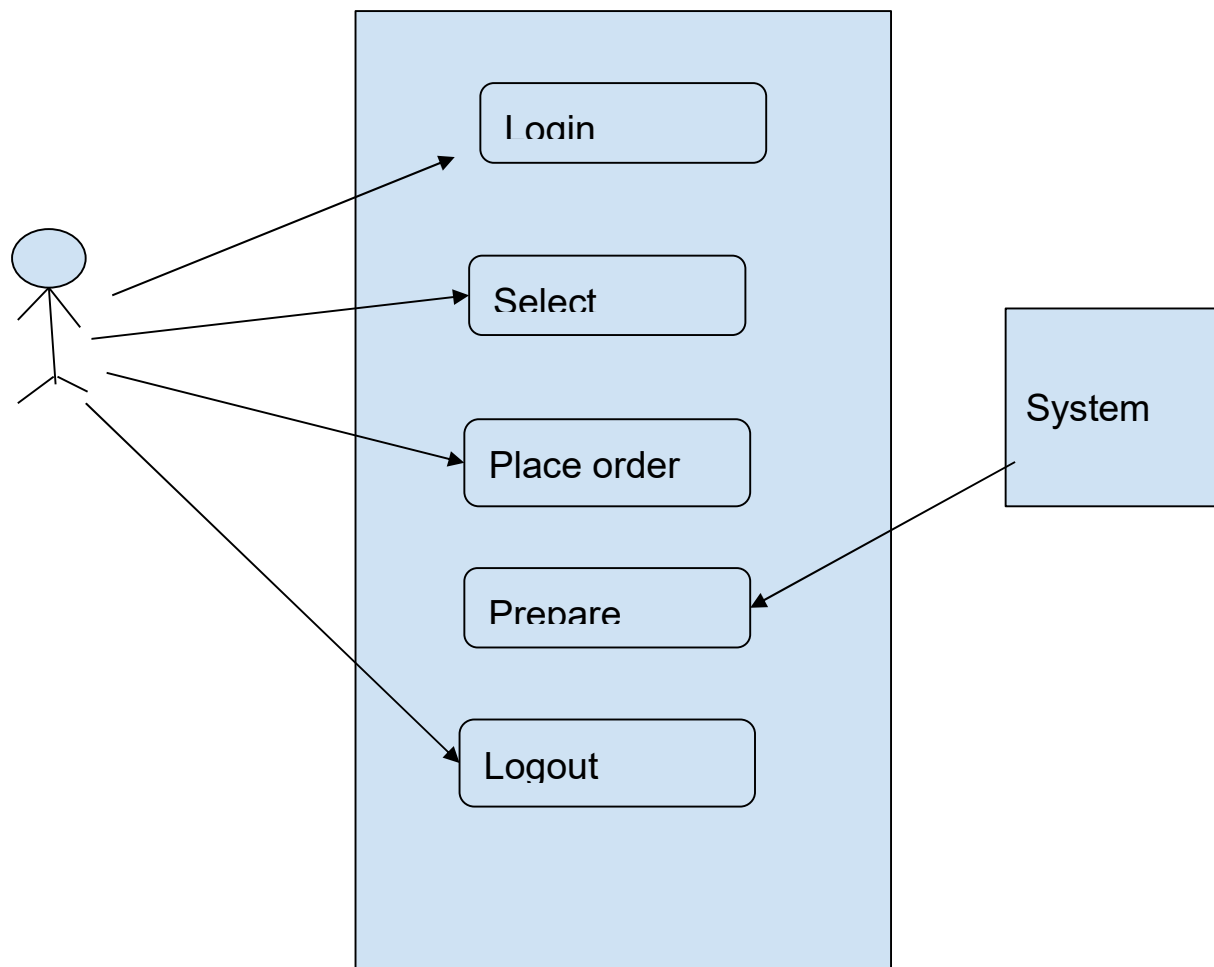
Use Case:- A set of scenarios that describing an interaction between user and system.



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

## 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 is also known as glass box testing, **structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focused on checking predefined inputs against

expected and desired outputs. It is based on the 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. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

### **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 time consuming when it comes to large-scale programming applications.
- White box testing is very 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**

--	--

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	<del>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.</del>
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 a decision point because there are two outcomes either true or false.

### **Difference 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.

### **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.

In this method, the tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all

functions if there are severe problems, then it is given back to the development team for correction.

### 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 Technique	Boundary Value Technique is used to test boundary values, 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 combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.
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.

<b>Error Guessing Technique</b>	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.
<b>UsCase technique</b>	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.

## 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 clientele 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.

### 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, APIs, 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 webpages are working correctly and make sure there are no broken links. Links to be checked will include -

- Outgoing links
- Internal links
- Anchor Links
- MailTo 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 webpages 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 webpages

### 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: NetMechanic

## 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 gzip 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.

## 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) MediumPriority- critical/major,minor
- d) Low Priority -trivial,enhancement

### 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.

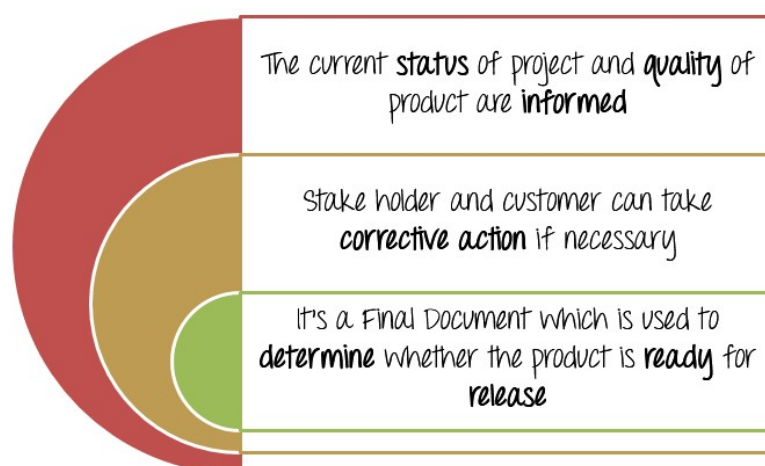
For example, if the test report informs that there are many defects remaining in the product, stakeholders can delay the release until all the defects are fixed.Test report is a **communication** tool between the Test Manager and the stakeholder. Through the test report, the stakeholder can **understand** the project situation, the quality of product and other things.

Test Report						
Test Cycle		System Test				
EXECUTED	PASSED				130	
	FAILED				0	
(Total) TESTS EXECUTED (PASSED + FAILED)						130
PENDING						0
IN PROGRESS						0
BLOCKED						0
(Sub-Total) TEST PLANNED (PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)						130

Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	High	
Edit Customer	Check Customer can be edited	100%	100%	0	High	
New Account	Check New account is added	100%	100%	0	High	
Edit Account	Check Account is edit	100%	100%	0	High	
Delete Account	Verify Account is delete	100%	100%	0	High	
Delete customer	Verify Customer is Deleted	100%	100%	0	High	
Mini Statement	Verify Ministatement is generated	100%	100%	0	High	
Customized Statement	Check Customized Statement is generated	100%	100%	0	High	

## Why Test Report?



### What does a test report contain?

Project Information	Test objective	Test summary	Defect
<ul style="list-style-type: none"><li>• Project Name</li><li>• Description</li></ul>	<ul style="list-style-type: none"><li>• Test Type</li><li>• Purpose</li></ul>	<ul style="list-style-type: none"><li>• Test Passed</li><li>• Test Failed</li><li>• Test Blocked</li></ul>	<ul style="list-style-type: none"><li>• Description</li><li>• Priority</li><li>• Status</li></ul>

### Interim report

It should provide detailed information that is relevant to the purpose of an interim report.

...

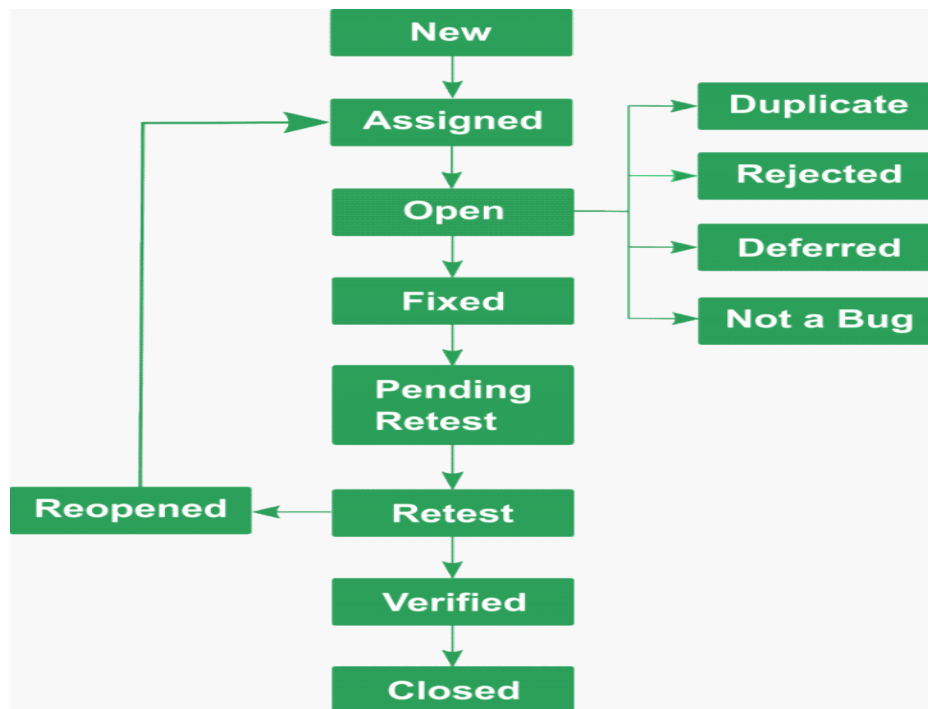
### Your interim report should:

- State your aims and objectives.
- Explain your research.
- Show what you have achieved.
- Demonstrate the steps to complete the project on time.

### 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:** Bug 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.