***zzzzzzzzzzzzzzzzManual Testing Notes***

***Testing or Software Testing:-***

Testing is an activity to detect and identify the defects in the software.

or

Testing is nothing but find out the bugs /defects /error in software.

The objective of testing is to release quality product to the client.

**Error/Bugs/ Defects in Software**:-

Error:- A mistake made by humans during coding is an error.

Defect:- An error find during the testing phase is called a defect.

Bugs:- A defect to be resolved by the development team is called a bug.

Failure:- When a build does not meet its specification then it is termed as Failures.

Faults:- A faults is an incorrect step, process, or data definition in a s/w product.

**Product or Project:-**

If software application is developed for specific customer based on requirement then it is called Project.

If software application is developed for multiple customers based on market requirement then it is called Product.

***Role of Tester:-***

* To test the software.
* To check the quality, correctness and completeness of the software.

**Quality Control:**

Quality Control is the actual testing of the software.

Quality Control focuses on testing for quality.

Quality Control is detecting defects.

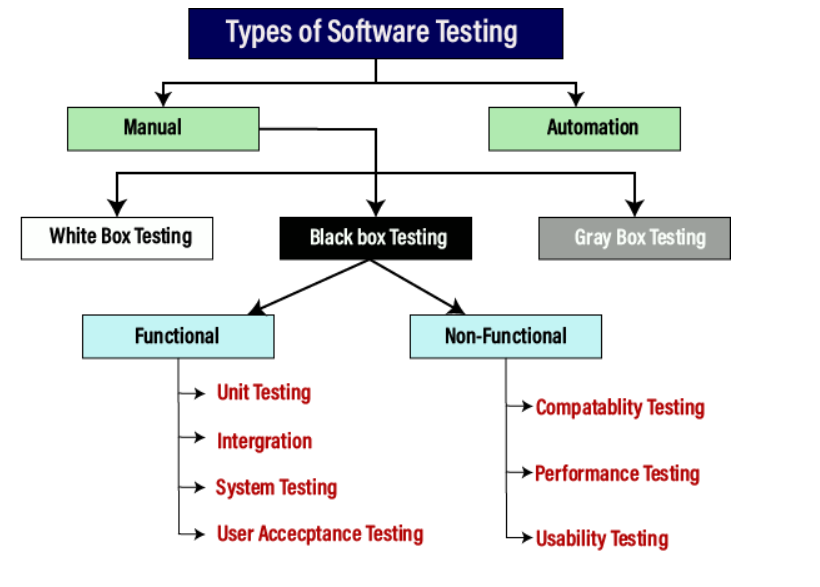
**Quality Assurance:**

Quality Assurance is Process Oriented.

QA focuses on building in quality.

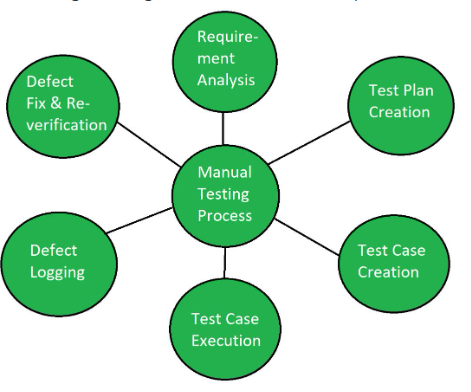
QA is preventing defects.

***Types of Software Testing:-***

***Manual Testing:-***

*T*he process of checking the functionality of an application as per the customer needs without taking any help of automation tools is known as manual testing**.**

In this testing, testers make test cases for the codes and test the software and give the final report about that software.



1. **White Box Testing:-**

* The white box testing is done by Developer.
* White-box testing is a method of software testing that tests internal structures or workings of an application, as opposed to its functionality.
* The purpose of implementing the white box testing is to emphasize the flow of inputs and outputs over the software and enhance the security of an application.
* White box testing is also known as **open box testing, glass box testing, structural testing, clear box testing, and transparent box testing**
* **Internal Structure means Condition, Loop, Operators, Syntax, Errors (Compile time error, Run time error, Logical Error).**

1. **Black Box Testing:-**

**Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths.

 Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known **as Behavioral Testing or Functional Testing.**

Black Box testing is done by tester because it is mainly focus on input and output of software.

Black box testing

* 1. **Functional Testing:-**

*Functional testing is a kind of black-box testing that is performed to confirm that the functionality of an application or system is behaving as expected.*

It is done to verify all the functionality of an application.

Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application under Test. The testing can be done either manually or using automation.

* + 1. **Unit Testing:-**

**Unit Testing** is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers.

 A unit may be an individual function, method, procedure, module, or object.

In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a White Box testing technique that is usually performed by the developer.

* + 1. **Integration Testing:-**

**Integration Testing** is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers

[Integration testing](https://artoftesting.com/integration-testing) is performed after unit testing. In integration testing, we test the group of related modules. It aims at finding interfacing issues between the module.

**Types of Integration Testing:-**

1. **Big-Bang Testing:- Big Bang Testing** is an Integration testing approach in which all the components or modules are integrated together at once and then tested as a single unit.

Convenient for small systems.

1. **Top-down Integration Testing –** In top-down integration, testing/integration starts from top modules to lower-level modules.

Takes help of stubs for testing.

1. **Bottom-up Integration Testing-** In bottom-up integration, testing starts from lower-level modules to higher-level modules up in the hierarchy.

It takes help of drivers for testing.

1. **Hybrid Integration Testing / Sandwich Testing -** Hybrid integration testing is the combination of both Top-down and bottom-up integration testing. In this approach, the integration starts from the middle layer, and testing is carried out in both the direction**.**

It makes use of both stubs as well as drivers.

**Stub and Driver:-**

**Stub**: Is called by the Module under Test.

**Driver**: Calls the Module to be tested.

**STUB:-**

In the case of top-down integration testing, many times lower-level modules are not developed while beginning testing/integration with top-level modules. In those cases, Stubs or dummy modules are used that simulate the working of modules by providing a hard-coded or expected output based on the input values.

**DRIVER:-**

In the case of bottom-up integration testing, drivers are used to simulate the

hierarchy.

**System Testing:-**

* Testing the overall functionalities of the application with respective client requirements.
* It is black box testing technique.
* This testing is conducted by testing team.
* Before conducting the system testing we should know the requirement.
* System testing focusses on below aspects;

User Interface Testing (GUI)

Functional Testing

Non- Functional Testing

Usability Testing.

**User Acceptance Testing:-**

After completion of system testing UAT team conducts acceptance testing in two levels

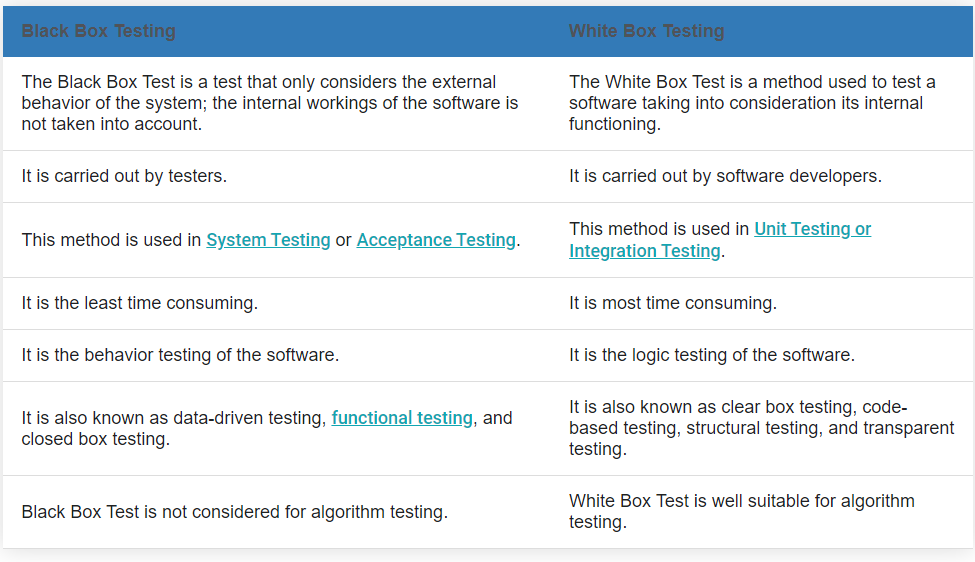
* Alpha Testing:-

**Alpha Testing** is a type of acceptance testing; performed to identify all possible issues and bugs before releasing the final product to the end users. Alpha testing is carried out by the testers who are internal employees of the organization

Critical issues and bugs are addressed and fixed immediately in Alpha Testing .

**Beta Testing** is performed by “real users” of the software application in “real environment” and it can be considered as a form of external [User Acceptance Testing](https://www.guru99.com/user-acceptance-testing.html). It is the final test before shipping a product to the customers.

Beta testing is the next phase, in which the software is tested by a larger group of users, typically outside of the organization that developed it.

**Difference between WhiteBox and BlackBox testing:-**

1. **Grey Box Testing:-**

**Gray Box Testing** is a software testing technique which is a combination of [Black Box Testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/) technique and [White Box Testing](https://www.geeksforgeeks.org/software-engineering-white-box-testing/) technique.

Gray box testing is a software testing technique to test a software product or application with partial knowledge of internal structure of the application.

**Testing Terminology:-**

**Adhoc Testing:-**

 When a software testing performed without proper planning and documentation, it is said to be Adhoc Testing.

Adhoc testing is also known as **Monkey testing and Gorilla testing** and Random Testing.

Ad-hoc testing is quite opposite to formal testing. It is an informal testing type. In Adhoc testing, testers randomly test the application without following any documents and test design techniques. This testing is primarily performed if the knowledge of testers in the application under test is very high. Testers randomly test the application without any test cases or any business requirement document.

**Adhoc Testing has two types:-**

1. **Buddy Testing**:- Buddy testing is done with at least two members. And one member is from the testing team, and another one is from the development team.

When the unit testing is performed on the application, then only we can perform buddy testing.

1. **Pair Testing:-** In this type of testing, two testers will work together to test the software, where they can share their ideas and identify the bugs or defects in the application.

One of them will test the application, and the other one can review and analyze the application.

**Regression Testing:-**

Regression testing is a type of [software testing](https://www.javatpoint.com/software-testing-tutorial). Test cases are re-executed to check the previous functionality of the application is working fine, and the new changes have not produced any bugs.

Regression means Re-test those parts of the application, which are unchanged.

**Re-Testing:-**

Testing Functionality repetitively is called Re-testing.

**Sanity Testing:-**

Sanity testing is a type of software testing that aims to quickly evaluate whether the basic functionality of a new software build is working correctly or not.

This is a basic functional testing conducted by test engineer whenever receive build( modules/Component) from development team.

It is entry level testing.

**Smoke Testing:-**

Smoke testing, also called **build verification testing or confidence testing**, is a software testing method that is used to determine if a new software build is ready for the next testing phase.

This is also basic functional testing conducted by developer or tester before release

the build to the next cycle.

**End to End Testing:-**

Testing the overall functionality of the system including the data integration among all the modules is called end to end testing.

**Exploratory Testing:-**

Exploring the application and understand the functionalities adding or modifying the existing test case for better testing is called Exploratory testing.

Exploratory Testing is widely used in Agile models.

Exploratory testing will be carried out by domain experts. They perform testing just by exploring the functionalities of the application without having the knowledge of the requirements.

**Globalization Testing:-**

Globalization testing is another type of software testing which is used to test the software that is developed for multiple languages, is called **globalization testing,** and improving the application or software for various languages is known as **globalization**.

This testing ensures that the application will support multiple languages and multiple features.

The purpose of Globalization testing is to ensure that software can be used internationally or worldwide. It is also called **Internationalization Testing**.

E.g-  In India, the **Google.com** supports most of the languages.

**Localization Testing:**-

**Localization Testing** is a software testing technique in which the behavior of a software is tested for a specific region, locale or culture. The purpose of doing localization testing for a software is to test appropriate linguistic and cultural aspects for a particular local.

**Positive Testing:-**

* **Positive Testing** is a type of testing which is performed on a software application by providing the valid data sets as an input.
* Testing conducted on the application in a positive approach to determine what system is supposed to do is called positive testing.
* Positive testing helps in checking if the customer requirements are justifying the application or not.

**Negative Testing:-**

**Negative Testing** is a testing method performed on the software application by providing invalid or improper data sets as input.

The purpose of negative testing is to ensure that the software application does not crash and remains stable with invalid data inputs.

**Non-Functional Testing:-**

**Non-Functional Testing** is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application.

An excellent example of non-functional test would be to check how many people can simultaneously login into a software.

1. **Compatibility Testing:-**

Compatibility Testing is a type of Software testing to check whether your software is capable of running on different hardware, operating systems, applications, browser, network environments or[Mobile](https://www.guru99.com/mobile-testing.html)devices.

Compatibility Testing is a type of Non-functional testing.

1. **Usability Testing:-**

To verify whether the application is user-friendly or not and was comfortably used by an end user or not. The main focus in this testing is to check whether the end user can understand and operate the application easily or not. An application should be self-exploratory and must not require training to operate it

**SDLC (Software Development Life Cycle)**

Software Development Life Cycle (SDLC) is a process used by the software industry to design, develop and test high quality softwares.

The SDLC aims to produce a high-quality software that meets or exceeds customer expectations, reaches completion within times and cost estimates.

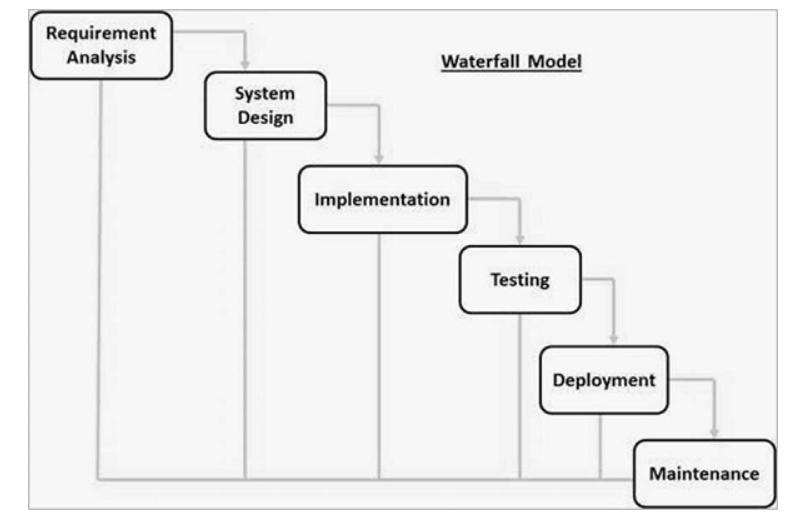


**Models in SDLC :-**

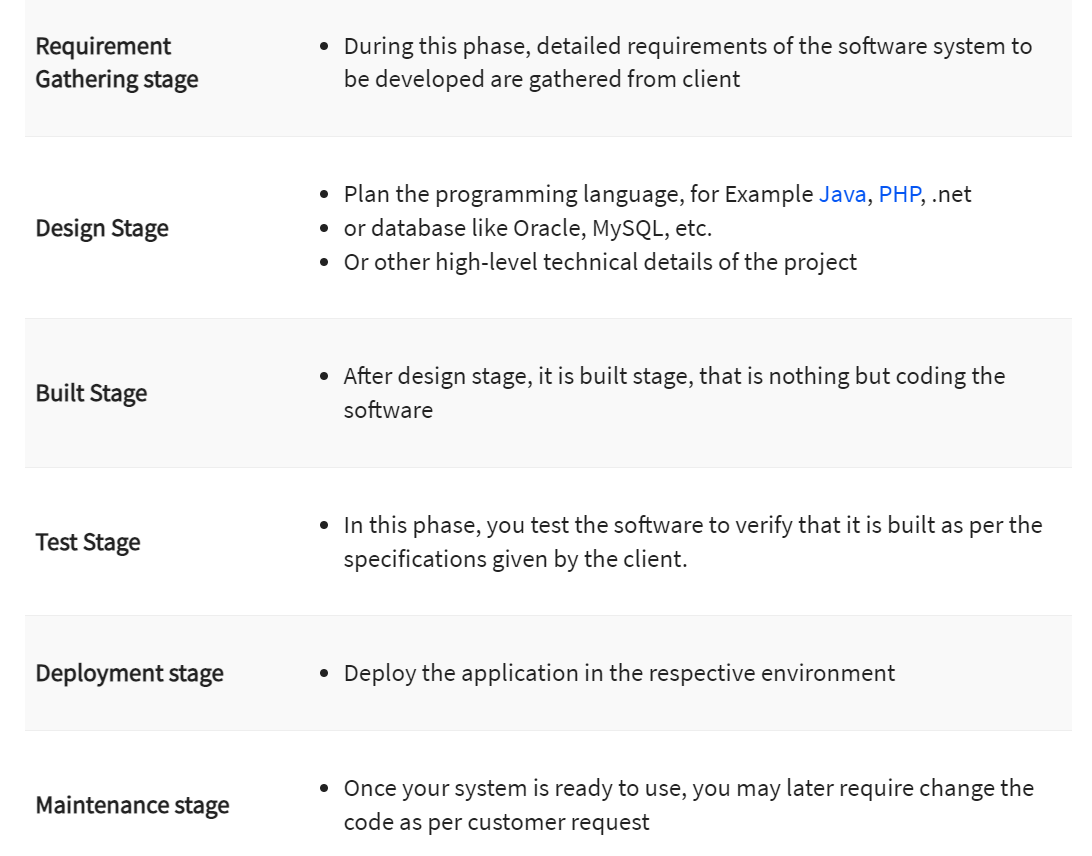
* Waterfall Model
* Spiral Model
* RAD Models
* V model
* Incremental Model
* Agile Model
* Prototype Model

**Waterfall Model:-**

**Waterfall Model** is a linear and sequential model that divides software development into pre-defined phases. Each phase must be completed before the next phase can begin with no overlap between the phases.

****

**flowing are the different waterfall Model:-**

****

**Advantage of Waterfall Model:-**

1. Quality of the product will be good.
2. Preferred for small Project where requirement are freezed.
3. Since requirement changes are not allowed, chances of finding bugs will be less.

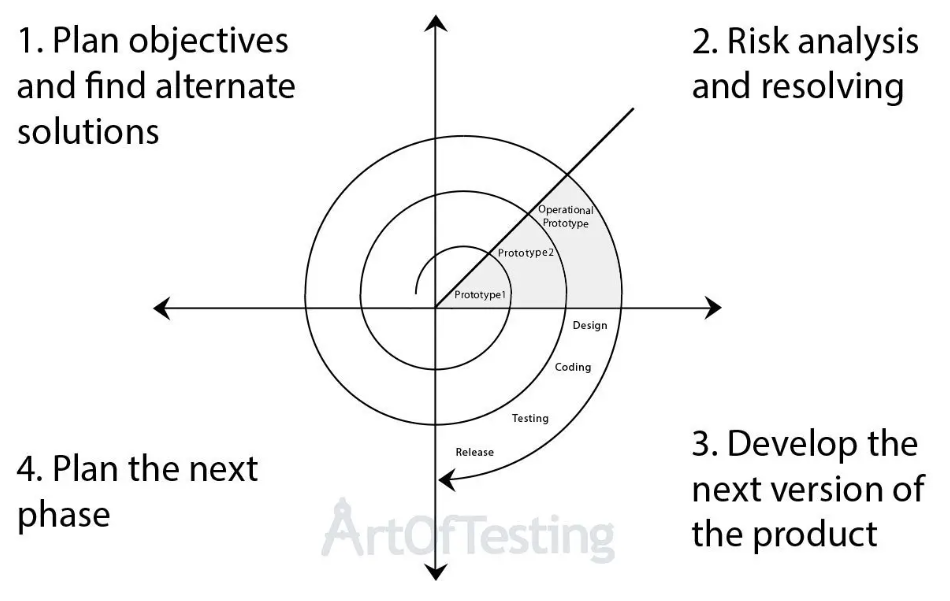
**Disadvantage of Waterfall Model:-**

1. Requirement changes are not allowed.
2. If there is defect in requirement that will be continued in later phases.
3. Testing will start only after coding.

**Spiral Model**:-

It is combination of Waterfall model, Iterative Model and Prototype Model.

The spiral model is a systems development lifecycle (SDLC) method used for risk management that combines the iterative development process model with elements of the Waterfall model. The spiral model is used by software engineers and is favored for large, expensive and complicated projects.



**Spiral Model Phases:-**

1. **Determine objectives and find alternate solutions –** This phase includes requirement gathering and analysis. Based on the requirements, objectives are defined and different alternate solutions are proposed.
2. **Risk Analysis and resolving –**In this quadrant, all the proposed solutions are analyzed and any potential risk is identified, analyzed, and resolved.
3. **Develop and test:** This phase includes the actual implementation of the different features. All the implemented features are then verified with thorough testing.
4. **Review and planning of the next phase –**In this phase,the software is evaluated by the customer. It also includes risk identification and monitoring like cost overrun or schedule slippage and after that planning of the next phase is started.

**Advantages of Spiral Models:-**

1. Testing is done is every cycle, before going to the next cycle.
2. Requirement changes are allowed after every cycle before going to the next cycle.
3. Customer will get to use the software for every module.

**Disadvantages of Spiral Models:-**

1. Requirement changes are not allowed between the cycle.
2. It is **not suitable for a simpler and smaller** project because of multiple phases
3. It requires **more documentation** as compared to other models.
4. Because of the prototype development and risk analysis in each phase, it is very **expensive and time taking**

**Prototype Model:-**

It is called dummy model.

Prototype testing means developers/ testers are checking if all the components mentioned in requirements are existing or not.

The requirements are collected from the client in a textual format. The prototype of the s/w product is developed. The prototype is just an image / picture of the required s/w product. The customer can look at the prototype and if he is not satisfied, then he can request more changes in the requirements.

We use this model when,

1) Customer is new to the s/w

2) When developers are new to the domain

3) When customer is not clear about his own requirement.

**Advantages of Prototype Models:-**

1. Reduce the risk of incorrect user requirement
2. Good where requirement are changing/uncommitted

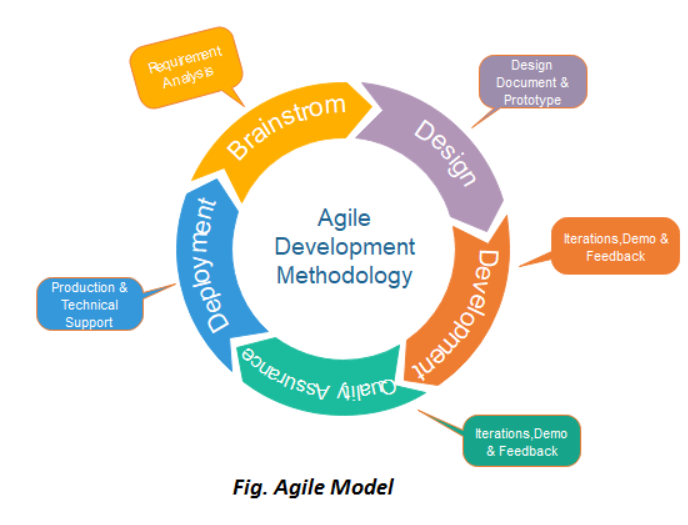
**Disadvantages of Prototype Models:-**

1. Special tools & techniques are required to build a prototype.
2. It is a time-consuming process.
3. Difficult to know how long the project will last.

**Agile Methodology:-**

The meaning of Agile is swift or versatile."**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.



**Phases of Agile Model:-**

1. Requirements gathering / Brainstrom
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback
   1. **Requirements gathering:** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.
   2. **Design the requirements:** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.
   3. **Construction/ iteration:** When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.
   4. **Testing:** In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
   5. **Deployment:** In this phase, the team issues a product for the user's work environment.

**6. Feedback:** After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.

**When to use the Agile Model:-**

* When frequent changes are required.
* When a highly qualified and experienced team is available.
* When a customer is ready to have a meeting with a software team all the time.
* When project size is small.

**Advantages of Agile Model:-**

1. Frequent Delivery
2. Face-to-Face Communication with clients.
3. Efficient design and fulfils the business requirement.
4. Anytime changes are acceptable.
5. It reduces total development time.

**Disadvantage of Agile Model**:-

1. Due to the shortage of formal documents, it creates confusion and crucial decisions taken throughout various phases can be misinterpreted at any time by different team members.
2. Due to the lack of proper documentation, once the project completes and the developers allotted to another project, maintenance of the finished project can become a difficulty.

**Agile Testing Methods:-**

* Scrum
* Crystal
* Dynamic Software Development Method(DSDM)
* Feature Driven Development(FDD)
* Lean Software Development
* eXtreme Programming(XP)

**SCRUM:-**

SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.

There are three roles in it, and their responsibilities are:

* **Scrum Master:** The scrum can set up the master team, arrange the meeting and remove obstacles for the process
* **Product owner:** The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
* **Scrum Team:** The team manages its work and organizes the work to complete the sprint or cycle.

### eXtreme Programming(XP):-

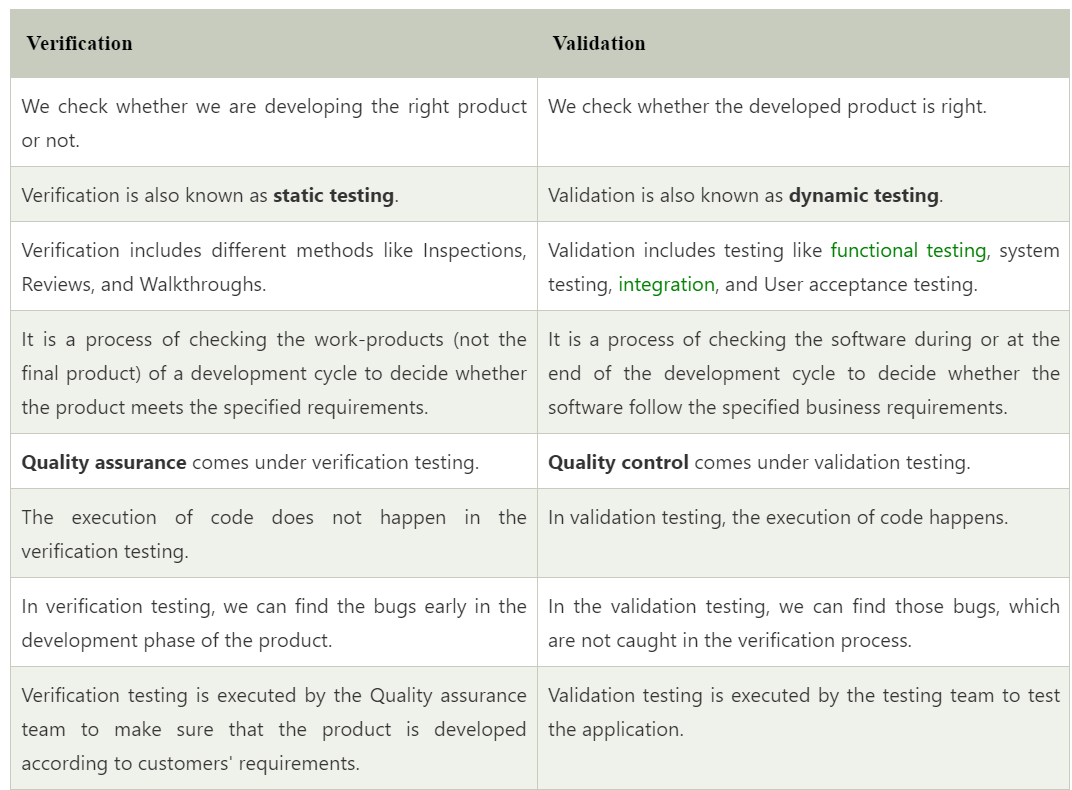
This type of methodology is used when customers are constantly changing demands or requirements, or when they are not sure about the system's performance.

**STLC(Software Testing Life Cycle):-**

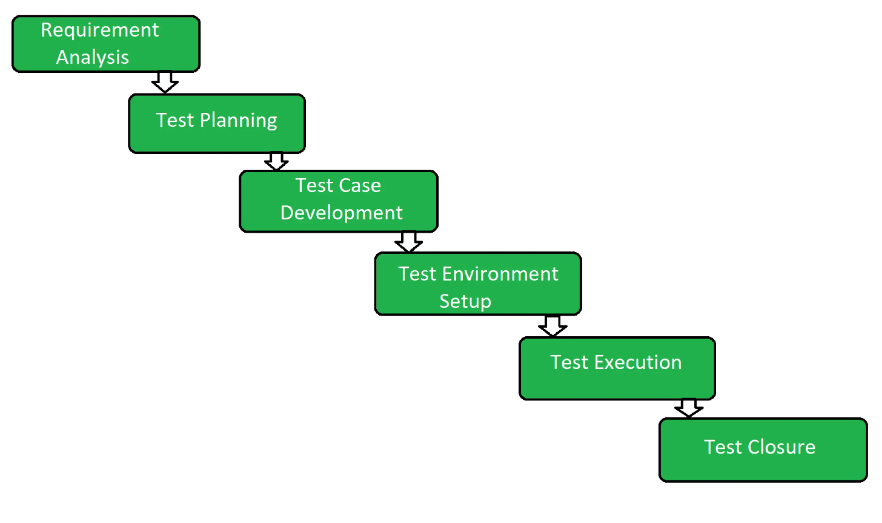
Software Testing Life Cycle (STLC) is a sequence of different activities performed during the software testing process.

Focus on Verification and Validation activities.

**Difference Between Verification and Verification**

****

Stages of STLC:-



**Requirement Analysis: (In the form of RTM, Automation Feasibility Report)**

In this phase quality assurance team understands the requirements like what is to be tested.

Quality Assurance team meets with the stakeholders to better understand the detail knowledge of requirement.

**Test Planning:- (Test Plan/ strategy document. Effort Estimation Document)**

Calculates estimated effort and cost for the testing work.

**Test Case Development:- (Test Cases/ Scripts, Test Data)**

Team note down the detailed test case.

Also prepare the required test data for testing

Test case Review.

**Test Environment Setup:- (Environment ready with test data set up**

**Smoke Test Results)**

Test Environment decides the conditions on which software is tested.

Testing Team is not involved. Either the developer or the customer creates the testing environment.

**Test Execution:- (Completed RTM with the execution status, Test Cases updated with results, Defects reports)**

Team start executing test cases based on prepared test cases in the earlier step.

**Test Closure:- (Test Closure Report, Test Metric)**

Last Stage of STLC in which the process of testing is analysed.

**Formal Testing and In-formal Testing:-**

**Formal testing** is a type of software testing in which the testing of the software is done with proper planning and with proper documentation. The degree of thoroughness and formality of test cases depends on the requirements of the project**.**

Formal testing can be applied to different levels of testing, such as unit testing, integration testing, system testing, or acceptance testing. Each level of testing has its own objectives and scope**.**

**Informal Testing**

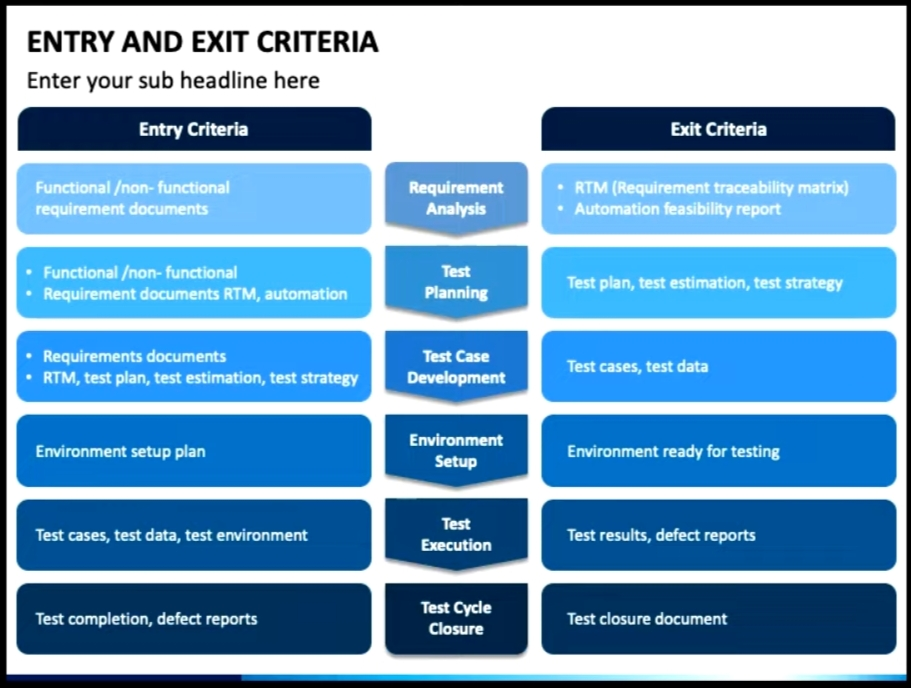
Ad hoc Testing is an informal or unstructured software testing type, it is done randomly and usually an unplanned activity that does not follow any documentation and test design techniques to create test cases.

**Entry and Exit Criteria in STLC:-**

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

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

**Difference Between Entry Criteria and Entry Criteria:-**



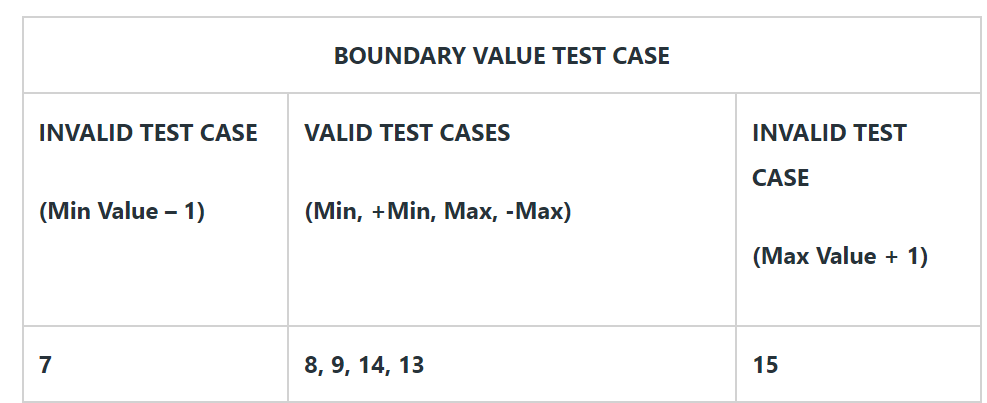
**Software Testing Technique:-**

Software Testing Techniques help you design better test cases.

1. [**Boundary Value Analysis**](https://www.geeksforgeeks.org/boundary-value-analysis-triangle-problem/)**:-**

[Boundary Value Analysis](https://www.geeksforgeeks.org/boundary-value-analysis-triangle-problem/) is based on testing the boundary values of valid and invalid partitions.

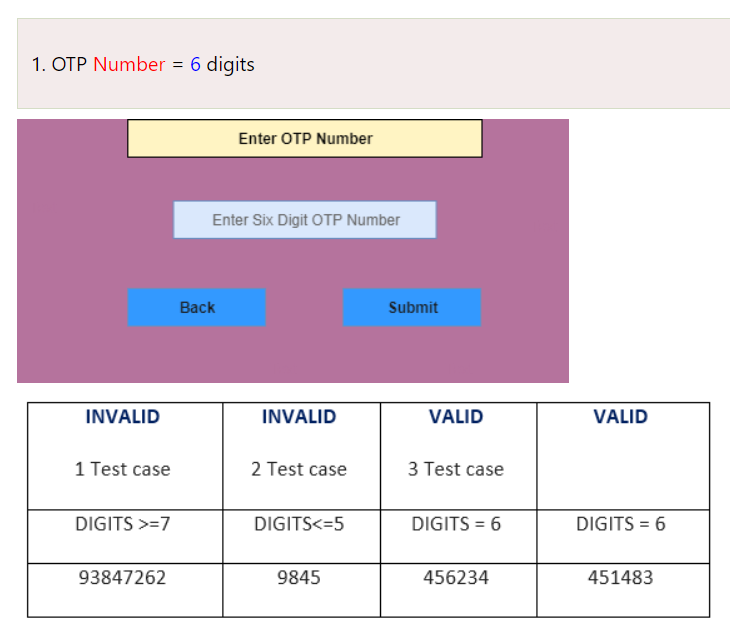
Every partition has its maximum and minimum values and these maximum and minimum values are the boundary values of a partition.

**Let us assume the next test case that takes the length of the input password from 8 to 14 characters**. 

1. **Equivalence Class Partition:-**

Equivalence Partitioning is also known as Equivalence Class Partitioning. In equivalence partitioning, inputs to the software or system are divided into groups that are expected to exhibit similar behavior, so they are likely to be proposed in the same way.

It helps to reduce the total number of test cases from infinite to finite. The selected test cases from these groups ensure coverage of all possible scenarios.

****

1. **State Transition :-**

**State Transition Testing** is a black box testing technique in which changes made in input conditions cause state changes or output changes in the Application under Test(AUT). State transition testing helps to analyze behaviour of an application for different input conditions. Testers can provide positive and negative input test values and record the system behavior.

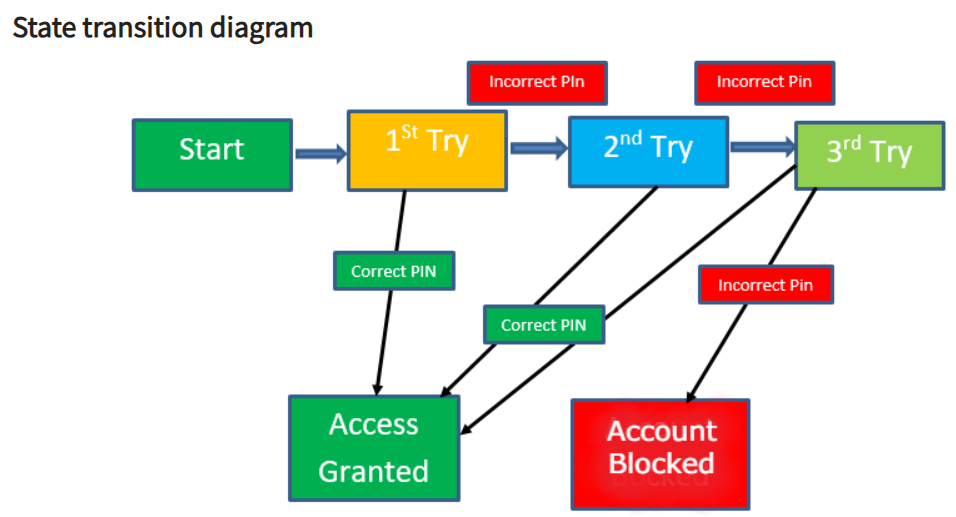
**When to Use State Transition?**

* This can be used when a tester is testing the application for a finite set of input values.
* When the tester is trying to test sequence of events that occur in the

application under test. I.e., this will allow the tester to test the application behavior for a sequence of input values.

### Example 1:

Let’s consider an ATM system function where if the user enters the invalid password three times the account will be locked.

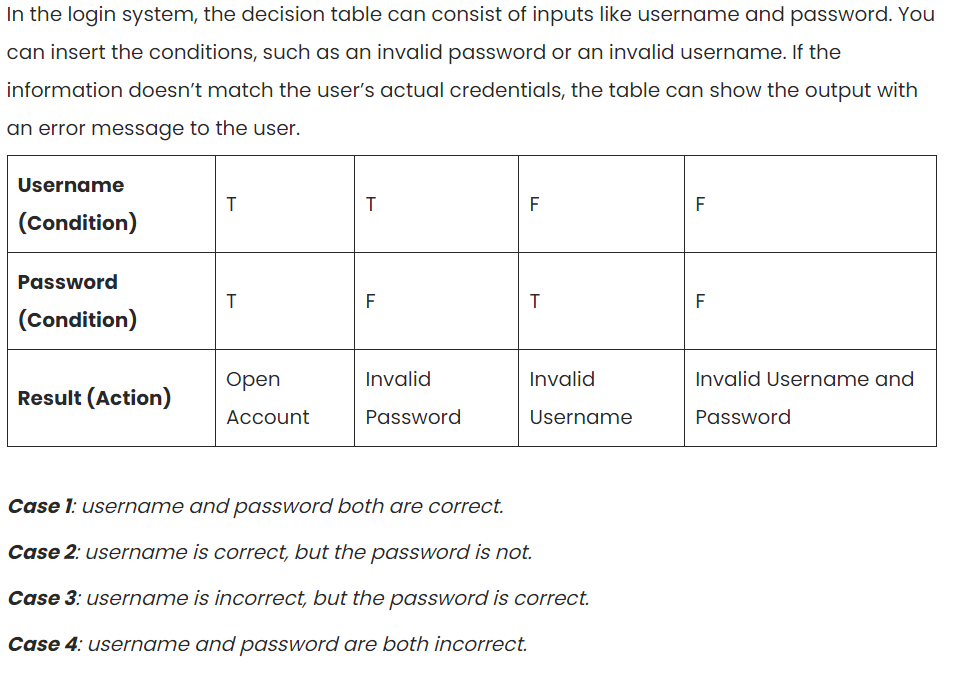
****

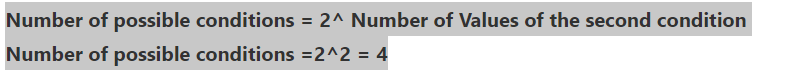
1. **Decision Table Testing :-**

Decision table testing is a [black box testing technique](https://artoftesting.com/black-box-testing) used to test multiple input combinations under different circumstances.

Decision tables are to as referred to as Cause-Effect tables, as they contain the causes and their respective effects in a table. They help in better [test coverage](https://artoftesting.com/test-coverage).

**Example:1**

****



**Error Guessing:-**

 Error guessing is a type of testing method in which prior experience in testing is used to uncover the defects in software. It is an experience based test technique in which the tester uses his/her past experience or intuition to gauge the problematic areas of a software application.

**Severity and Priority:-**

Severity: The impact of the bug on the application is known as severity.

It can be a **blocker, critical, major, and minor** for the bug.

**Blocker:** if the severity of a bug is a blocker, which means we cannot proceed to the next module, and unnecessarily test engineer sits ideal.

There are two types of **blocker** bug, which are as follows:

**A major feature is not working:** Login to HDFC, amount transfer is not working

**The major flow is not working:** Login and signup itself not working in HDFC application.

**Critical:** if it is critical, that means the main functionality is not working, and the test engineer cannot continue testing.

**Major:** if it is major, which means that the supporting components and modules are not working fine, but test engineer can continue the testing.

**Minor:** if the severity of a bug is major, which means that all the U.I problems are not working fine, but testing can be processed without interruption.

**Priority:-**

Priority is a term that defines how fast we need to fix a defect. Higher the priority the sooner the defect should be resolved.

[Types of Priority of bug/defect can be categorized into three parts :](https://www.guru99.com/software-testing-introduction-importance.html)

* **[Low:](https://www.guru99.com/software-testing-introduction-importance.html)**[The Defect is an irritant but repair can be done once the more serious Defect has been fixed](https://www.guru99.com/software-testing-introduction-importance.html)
* **[Medium:](https://www.guru99.com/software-testing-introduction-importance.html)**[During the normal course of the development activities defect should be resolved. It can wait until a new version is created.](https://www.guru99.com/software-testing-introduction-importance.html)
* **[High:](https://www.guru99.com/software-testing-introduction-importance.html)**[The defect must be resolved as soon as possible as it affects the system severely and cannot be used until it is fixed.](https://www.guru99.com/software-testing-introduction-importance.html)
* **Critical/Immediate (P1):** This has to be fixed immediately within 24 hours. This generally occurs in cases when an entire functionality is blocked and no testing can proceed as a result of this.

The different levels for both Priority and Severity.

* High Priority, High Severity
* High Priority, Low Severity
* High Severity, Low Priority
* Low Severity, Low Priority
* **High Priority, High Severity:-**
  1. The products added to the cart of an e-commerce website are not visible on the payment page.
  2. The login button of the application is not working.

* **High Priority, Low Severity:-**
  1. The logo of the company’s welcome page is distorted.
  2. The action buttons are not visually appealing or the information on the page appears hazy.

* **High Severity, Low Priority:-**

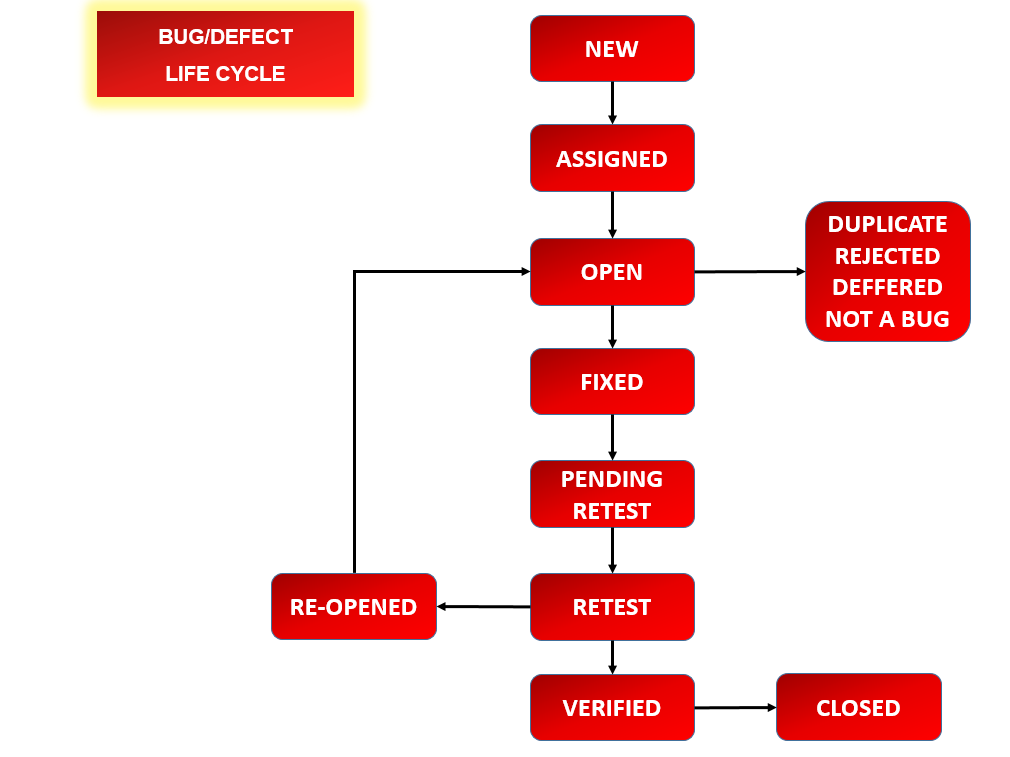
1. If the application is crashing on passing very large input for processing (which is very rarely done).
2. There are some buttons on the website which are overlapping. Although clickable, are creating a fuss.

* **Low Severity, Low Priority:-**

1. A spelling mistake on the page of the site which is not frequently visited.
2. The color of any text does not match the theme of the website.

**Bug/ Defect Life Cycle:-**

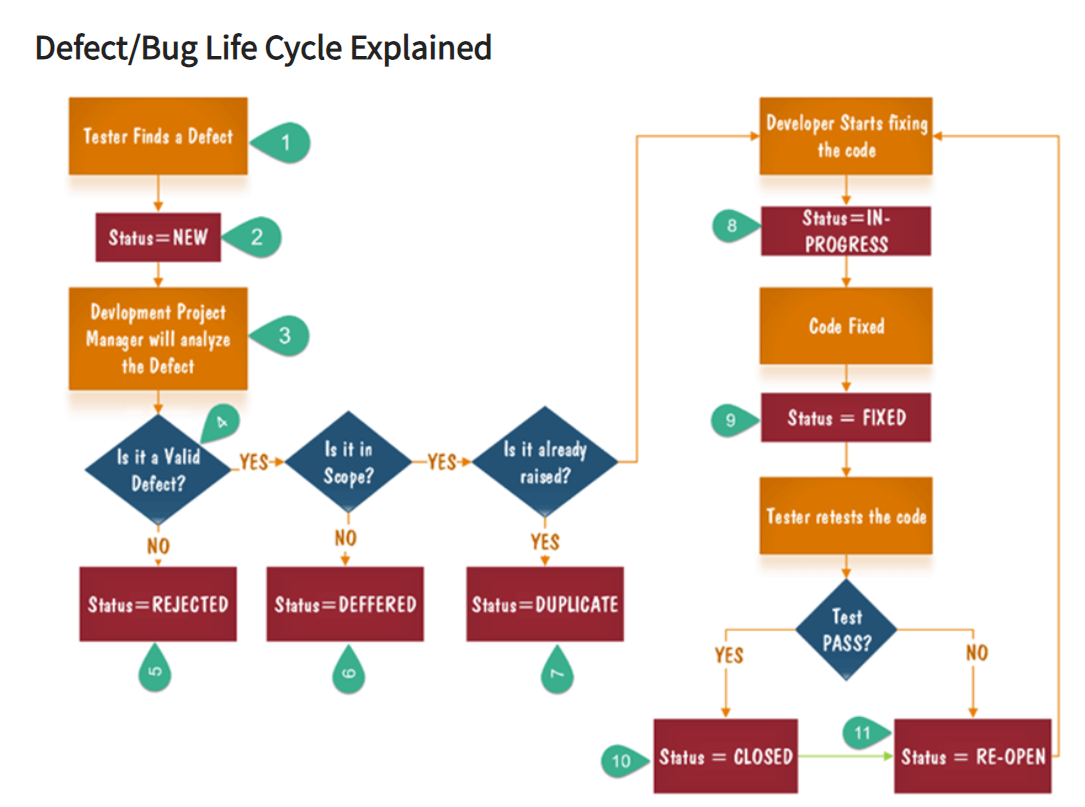
The Defect Life Cycle, also known as the Bug Life Cycle, is a cycle of defects from which it goes through covering the different states in its entire life. This starts as soon as any new defect is found by a tester and comes to an end when a tester closes that defect assuring that it won’t get reproduced again.



## Defect States Workflow:-

The number of states that a defect goes through varies from project to project. Below lifecycle diagram, covers all possible states

* **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
* **Open**: The developer starts analyzing and works on the defect fix.
* **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as “Fixed.”
* **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the [software testing](https://www.guru99.com/software-testing-introduction-importance.html) remains pending from the testers end, the status assigned is “pending retest.”
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to “Re-test.”
* **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is “verified.”
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to “reopened”. Once again the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status “Closed.”
* **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to “duplicate.”
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to “rejected.”
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status “Deferred” is assigned to such bugs
* **Not a bug**: If it does not affect the functionality of the application then the status assigned to a bug is “Not a bug”.



1. Tester finds the defect
2. Status assigned to defect- New
3. A defect is forwarded to Project Manager for analyze
4. Project Manager decides whether a defect is valid
5. Here the defect is not valid- a status is given “Rejected.”
6. So, project manager assigns a status **rejected**. If the defect is not rejected then the next step is to check whether it is in scope. Suppose we have another function- email functionality for the same application, and you find a problem with that. But it is not a part of the current release when such defects are assigned as a **postponed or deferred**status.
7. Next, the manager verifies whether a similar defect was raised earlier. If yes defect is assigned a status **duplicate**.
8. If no the defect is assigned to the developer who starts fixing the code. During this stage, the defect is assigned a status **in- progress.**
9. Once the code is fixed. A defect is assigned a status **fixed**
10. Next, the tester will re-test the code. In case, the[Test Case](https://www.guru99.com/test-case.html)passes the defect is **closed.** If the test cases fail again, the defect is **re-opened** and assigned to the developer.
11. Consider a situation where during the 1st release of Flight Reservation a defect was found in Fax order that was fixed and assigned a status closed. During the second upgrade release the same defect again re-surfaced. In such cases, a closed defect will be **re-opened.**

**Test Plan:-**

Test Plan is A document describing the scope, approach, resources, and schedule of intended test activities.

**Test Plan is a dynamic document**

Test Plan is more or less like **a blueprint of how the testing activity is going** to take place in a project.

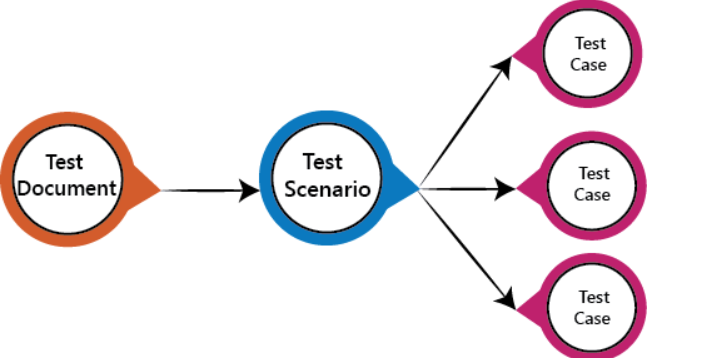
**Importance of Test Plan:-**

* 1. **Gives clarity.**
  2. **Help you to have better estimates.**
  3. **You can present to stake what you are going to do and what you are not going to do.**
  4. **Clarity over QA strategy.**
  5. **Mitigate the Risk**
  6. **Help you to track the testing.**

**Test Scenario:**

A **Test Scenario** is defined as any functionality that can be tested. It is also called *Test Condition* or *Test Possibility*.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements.



**Test Plan:-**

A **Test Case** is a set of actions executed to verify a particular feature or functionality of your software application.

A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement.

The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

**Requirement Traceability Matrix(RTM) :-**

Traceability matrix is a table type document that is used in the development of software application to trace requirements.

It can be used for both forward (from Requirements to Design or Coding) and backward (from Coding to Requirements) tracing.

It is also known as **Requirement Traceability Matrix (RTM) or Cross Reference Matrix (CRM).**

**The test engineer** will prepare RTM for their respective assign modules, and then it will be sent to the Test Lead.