



MANUAL TESTING NOTES

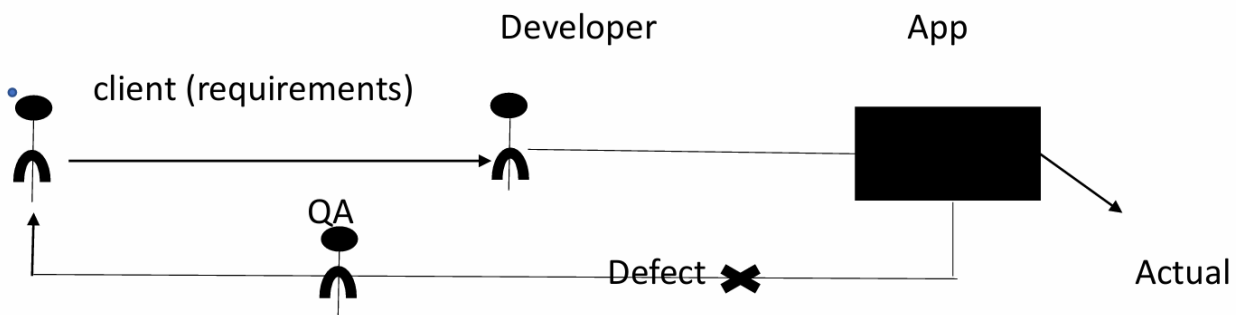
Software Testing

Software: A set of executable programs or collection of programs in a computer is called Software.

Testing: Comparison between expected values and actual values.

- Before developing the application, the client requirements are called Expected values.
- After development, the final output of the application is called the Actual value.

Definition: Software Testing It is a process of executing an application to find defects.



Quality: Customer/Client/End-user satisfaction is called Quality.

Difference between Error, Defect, Bug, and Failure:

Error: While developing the application if the programmer finds any mistake in the coding part, it is called Error.

Defect: During testing, if any expected value is not equal to the actual value, it is called a Defect.

Bug: Defect accepted by the development team then called a Bug/Anomaly.

Failure: After development and testing, during utilization if a customer faces any problem is called Failure.

Difference between Project and Product:

Project: An application developed for a specific client requirement is called a Project.

Ex: TCS, and Tech Mahendra companies are examples of project/service-based companies.

Product: An application developed for multiple client requirements or entire market requirements is called a Product.

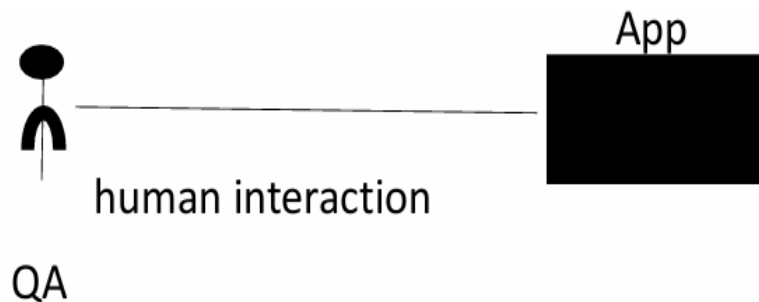
Ex: Gmail, Yahoo mail, Rediff mail etc. Google and Microsoft companies are examples of product-based companies.

Software Testing

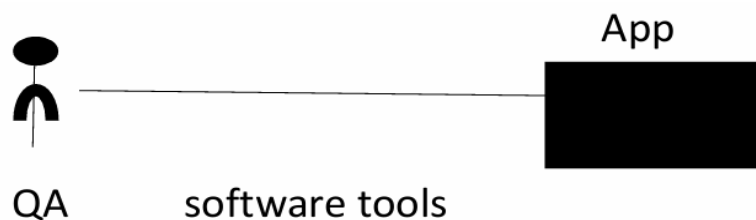
1)Manual Testing

2)Automation Testing

Manual Testing: It is a process of testing an application by human interactions is called Manual Testing.

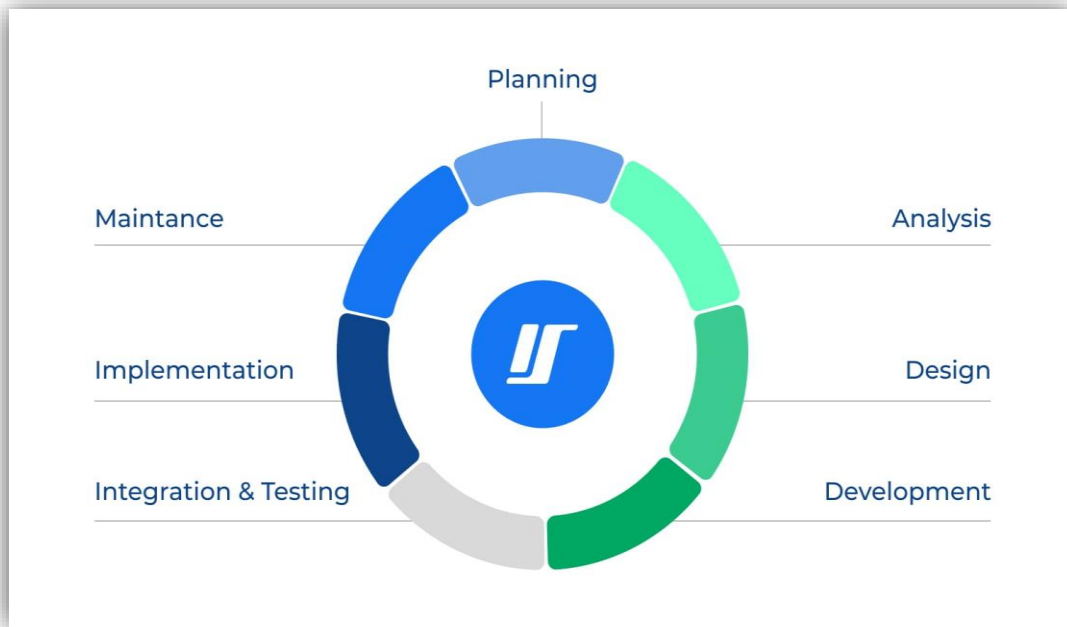


Automation Testing: App It is a process of testing an application with the help of software tools called Automation Testing.



SDLC

- SDLC stands for Software Development Life Cycle. It will explain all the implementation activities of project development.
- SDLC aims to produce high-quality software that meets customer expectations.



- 1) **Planning:** Create a basic plan for the application based on business requirements

Task: Prepare a plan for project implementation.

Role: Project Manager with the help of the development manager.

Documents: The project plan contains the development plan, Testing plan, and Design plan documents.

- 2) **Analysis:** Translate plans and goals into actionable ideas

- 3) **Design:**

Task: Requirements designing for application development.

Role: Architect/ Technical team

Document:

- ✚ HLD [High-Level Design]

- ✚ LLD [Low-Level Design]

Process: Based on requirements and plan, before starting coding by the programmer, the Architect / Technical team designs the requirements as HLD / LLD documents.

- ✚ HLD defines the requirements as a module-wise.

- ✚ LLD defines the requirements as functionalities.

- 4) **Implementation (or) Coding:** Developers develop the application and do the white box testing. It is collectively of 2 levels –

- ✚ Unit Testing and

- ✚ Integration Testing.

- 5) **Testing:** Whenever 100% coding and White Box Testing are completed, to confirm the correctness of the customer's requirements, testing a front-end application is called "Black Box Testing".

It is collectively of 2 levels –

- ✚ System Testing and

- ✚ User Acceptance Testing.

- 6) **Deployment:** Move the application into production

Once all activities are completed, if the client is also accepted in user acceptance testing, then the delivery team will release the project to the customer.

- 7) **Maintenance:** Monitor the application for issues

After delivery to the customer, during utilization, if the customer is facing any problem called "Failure." If any failure occurs, to handle those requirements, the Maintenance team will provide support to the project.

Difference between Verification and Validation

Verification: It is a process of verifying whether developing the project is right. It is called “Verification,” also called “Static testing.”

Validation: It is validating whether the developed product is right. It is called “Validation,” also called “Dynamic Testing.”

Software Development Models

Various software development approaches have been defined and designed and are used during the software development process.

Those approaches are referred to as “Software Development Models.”

Software Development Models are classified into two

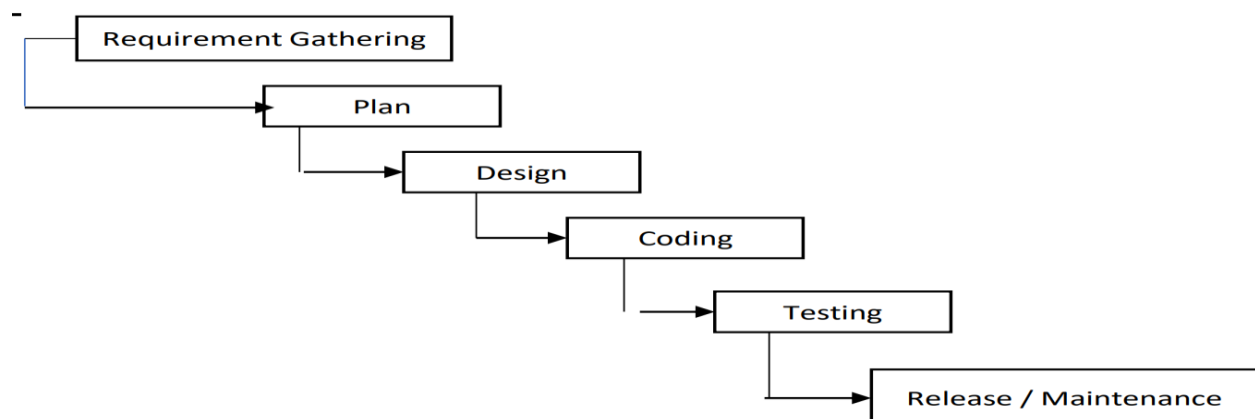
- 1) Sequential Models
- 2) Incremental Models

1. Sequential Models

These models are best suitable for the small size of projects, where all SDLC activities will be carried out one after another, for the entire project.

“The waterfall model and V model are the best examples of sequential models.”

Waterfall Model:



It is a sequential model, suitable for small size of projects.

- The waterfall model same as the SDLC process.
- Whenever client requirements are clear in the Waterfall model all implementation activities will be carried out step by step process.
- In the Waterfall model, only Validation is required, verification is not required.

In this model, the flow of activity looks like a waterfall, so this model is titled the Waterfall model.

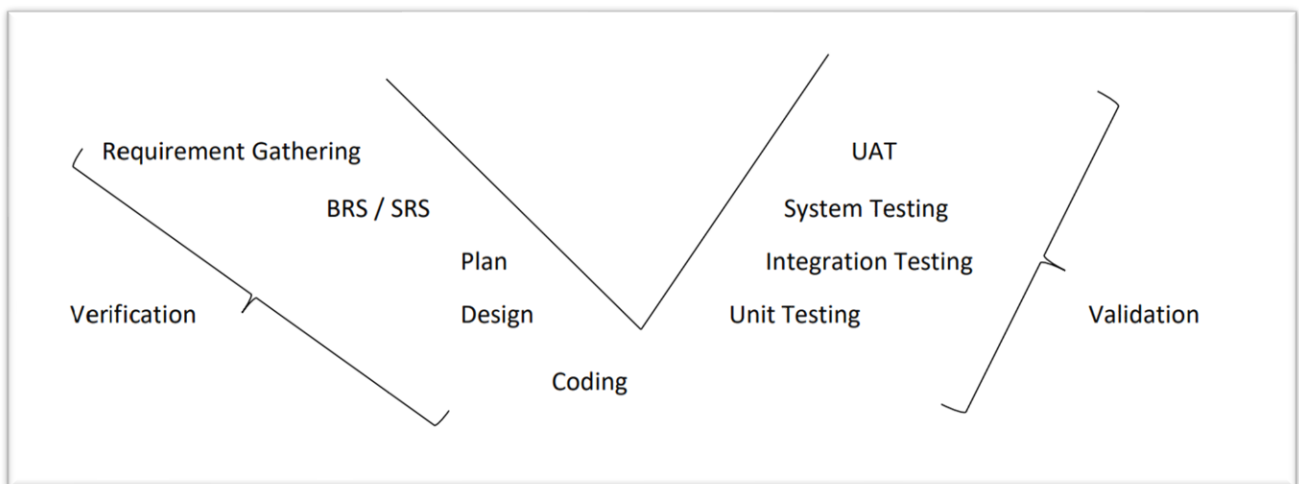
Advantages: -

- Easy to understand and simple.
- Works well for small projects.
- Easy to manage step-by-step activities.

Disadvantages: -

- Not suitable for big-size projects.
- In the middle of the project, if any change requirements are needed then going back and changing requirements are tedious job.

V Model:



‘V’ stands for Verification and Validation. This model is suitable for the small size of applications, where the requirements are clear and chances of making mistakes are higher, while implementing

the application to reduce this at every step of implementing software testing apply both verification and validation.

In the 'V' model, once requirement documents are ready, based on those requirements, along with developers, testers will start testing activities such as Test plan, Test Scenario, and Test case documentation. So, testing activities will start before coding.

Advantages:

- Simple and easy to use.
- Save time.
- Testing activities will start before coding.
- Ensure 100% quality.
- Implementing both verification and validation.

Disadvantages:

- Not suitable for the big size of the projects.
- If any changes happen in the middle of the project, requirement, development, and testing documents must be updated.

2. Incremental Models

These models are best suitable for big-size projects. In the incremental model, a big project will be divided into modules, then all SDLC activities will be carried out module by module.

RAD model, Prototype model, Spiral model, and Agile model are the best examples of Incremental models.

1. RAD (Rapid Application Development) Model

2. Prototype Model

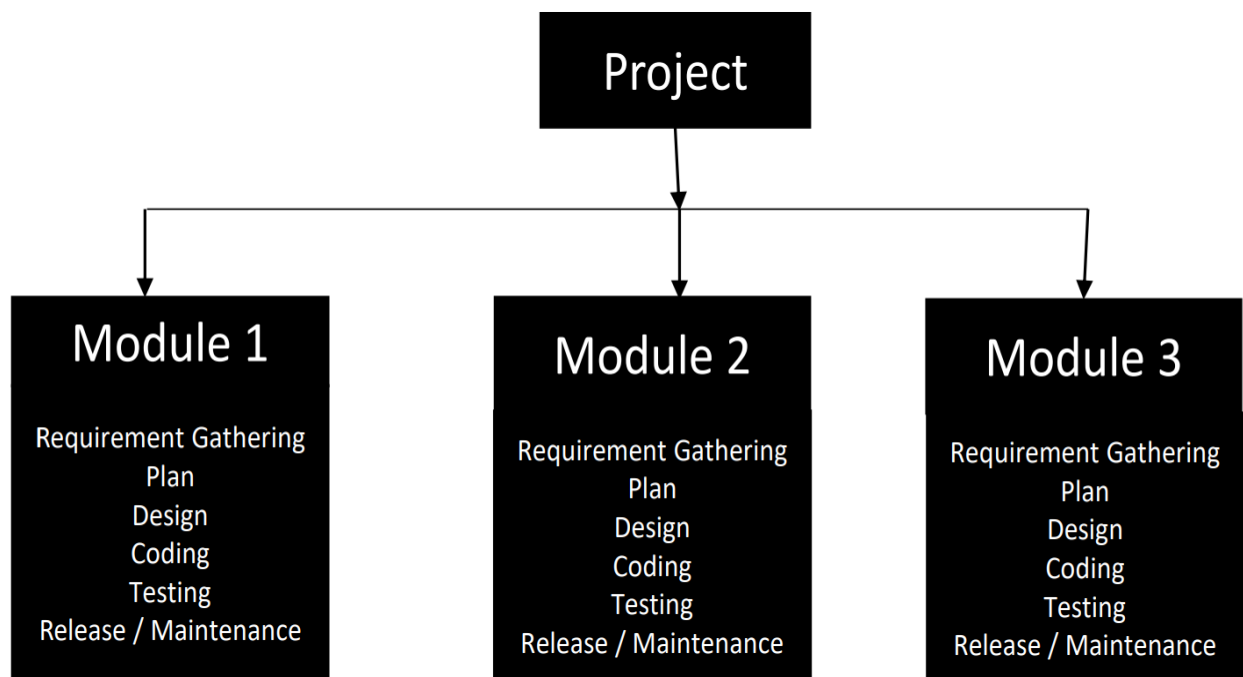
3. Spiral Model

4. Agile Model

1. RAD Model:

RAD stands for “Rapid Application Development.” Due to lack of time or within the short term to complete big-size projects in the RAD model, a big project will be divided into modules and every module will be considered as a mini project, a separate team will be scheduled to implement all SDLC activities, for those modules parallelly.

Once all modules are implemented, these modules will be combined and delivered to the customer.



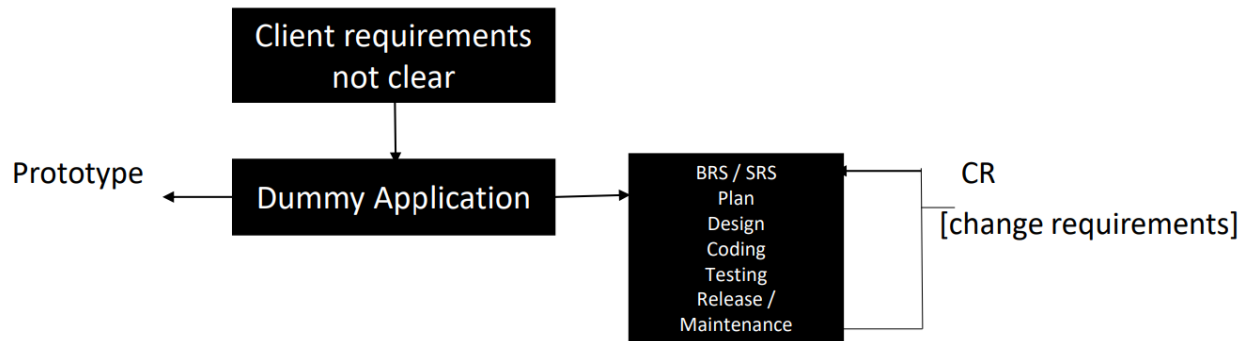
Advantages:

- Reduce development time.
- Suitable for big size of projects.

Disadvantages:

- Not suitable for the small size of projects.
- Expensive model.
- Integration problems will occur.

2. Prototype Model:



Due to a lack of requirements or if customers are confused about the requirements, in this model instead of developing an actual application, develop a Dummy application called a “Prototype “. It will be accepted by the client.

Once the client approves the Prototype, based on the sample BRS/SRS will be designed, and based on those requirements all implementation activities carried out such as Plan, Design, Coding, and Testing activities. If any changes are requested by the customer after delivery, the same will be implemented as a Change Requirement (CR). Those cycles will be continued with the same process.

Advantages:

- Change Requirements will be implemented after delivery to the customer.
- If customer requirements are not clear also, we can implement the project based on prototype.
- Missing functionality can be identified easily.

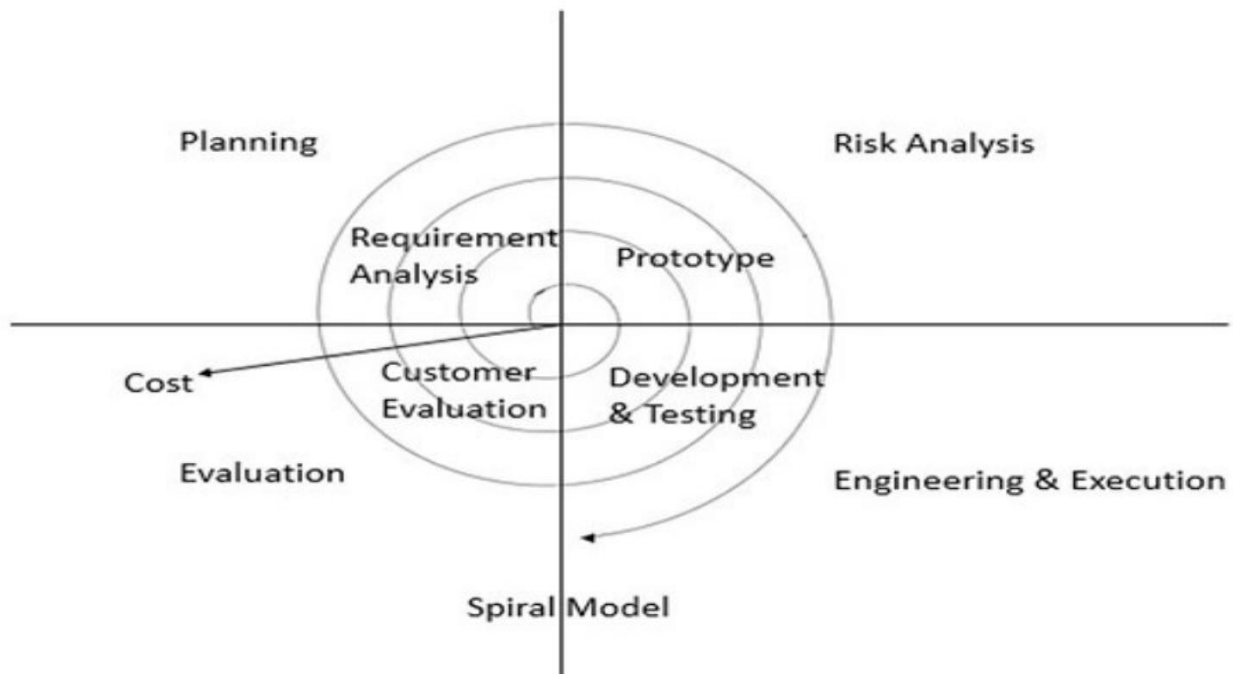
Disadvantages:

- Not suitable for the small size of projects.
- Expensive model.
- More time and resources are required.

3. Spiral Model:

This model is suitable for Maintenance or Long-term projects. Whenever customer requirements are frequently changing or dynamic requirements of the customer, in this model application will be implemented requirement by requirement.

In this model, the flow of activity looks like a spiral net, so this model is titled a “Spiral Model.”



Advantages:

- Additional functionality can be added in the next cycles.
- The project will be delivered to customers early in the software cycle.
- Best suitable for Maintenance and Long-term projects.

Disadvantages:

- Expensive model.
- Require more time.
- Not suitable for short-term or small-size projects.

4. Agile Model/ Agile Methodology:

The agile model is an **Iterative** (the same process will be repeated and again multiple times, which means getting requirements, analysis, design, coding, and testing will repeat) and **Incremental** (adding new features/modules on existing software) process or approach.

Agile Principles:

- Customer no need to wait for a long time.
- We develop, test, and release pieces of software to customers with a few numbers of features.
- We can accommodate/accept requirement changes from customers easily.
- In this model, work along with the client representative team from day 1 onwards.
- Release is very fast within 2/3 weeks can release product/project.

It has more frameworks, such as XP, Scrum, and Iterative, where we will discuss on Scrum process.

Scrum:

Scrum is a framework through which software product is built by following Agile principles.

Agile is a defined process with some principles.

Scrum includes a group of people called Scrum Team (normally 5 to 9 members).

They are the Product Owner, Scrum Master, Development (Dev) team, and Testing (QA) team and everybody works together to deliver quality products within a short time.

Roles and Responsibilities of Scrum Team:

1. Product Owner:

- The Product Owner is the main person who writes or defines the functionality of the product by directly contacting with client and getting the requirements.
- He will prioritize those features and assign them to developers and testers.
- Accept or reject work results.

2. Scrum Master:

- Scrum Master takes care of the entire Agile process (from the beginning of software to delivery).
- The Project Manager or Client will work as a Scrum Master.
- He is responsible for the sprint plan, sprint meeting, and scrum meeting.
- He will handle any issues in the team.
- He will give awareness to people on the Agile process.

3. Dev Team: Developers will develop the software.

4. QA Team: Testers will test the software.

Scrum Terminology/Agile/Scrum Ceremonies

1) User Story:

It is a feature/module in software. It gives a general explanation of software features.

Note:

The Product Owner will define these User Stories and Epic.

2) Product Backlog:

It is a list of all user stories prepared by the Product Owner at the beginning of the Agile process.

3) Sprint:

Duration of time to complete User Stories, decided by Product Owner and Team, usually 2-4 weeks.

4) Sprint Planning Meeting:

This meeting is conducted by the team (Product Owner, Scrum Master, Dev team, Testing team), to define what can be delivered in the sprint / particular duration of time and to review previous user stories completed as per the sprint plan or not and if any user stories are unable to complete within sprint plan, then it will consider as backlogs and to explain new sprint user stories.

5) Sprint Planning Meeting:

Meeting conducted by team (Product Owner, Scrum Master, Dev team, Testing team), to define what can be delivered in the sprint / particular duration of time and to review previous user stories completed as per sprint plan or not and if any user stories are unable to complete within sprint plan, then it will consider as backlogs and to explain new sprint user stories.

6) Sprint Backlog:

List of committed stories by Dev/QA for specific sprints.

7) Scrum Meeting:

Scrum call/standup meeting conducted by the Scrum Master every day 15 mins, called as Standup meeting/scrum call, to discuss the status of the project.

Every day it focuses on 3 things in this meeting:

- a. What did you do yesterday?
- b. What will you do today?
- c. Are there any blockers in your way?

8) Sprint Retrospective Meeting:

It is conducted only once at the end of every sprint. It focuses on 3 things:

- a) What went well?
- b) What went wrong?
- c) What steps need to be taken?

9) Story Point:

A rough estimation of user stories given by Dev and QA. During the sprint planning meeting itself, every story will be estimated by the Product Owner, Dev, and QA.

1 story point = 1 hr/ 1 day (6-8 hrs) depending on company

Suppose, for

Login -> Dev given 5 (means 5 hrs)

QA given 3 (means 3 hrs),

so total 5+3 =8 hrs (means 1 day)

10) Burndown/Up Chart:

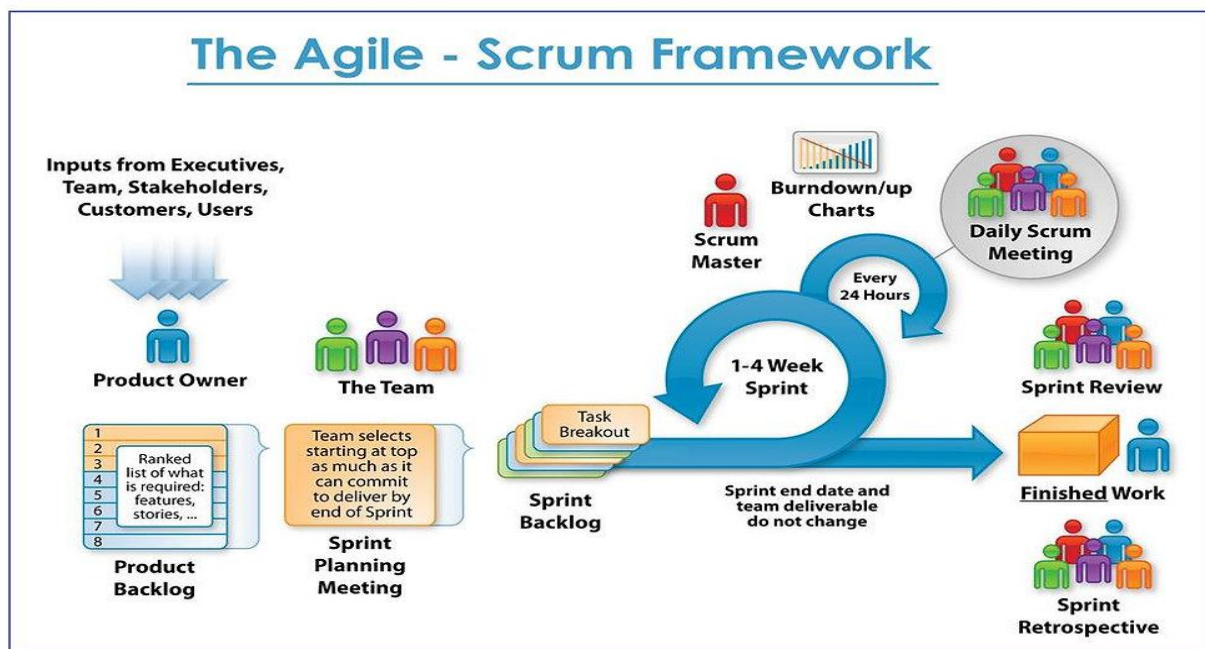
It is a graph that shows how much work remains in the sprint. The ScrumMaster will design this daily.

Advantages:

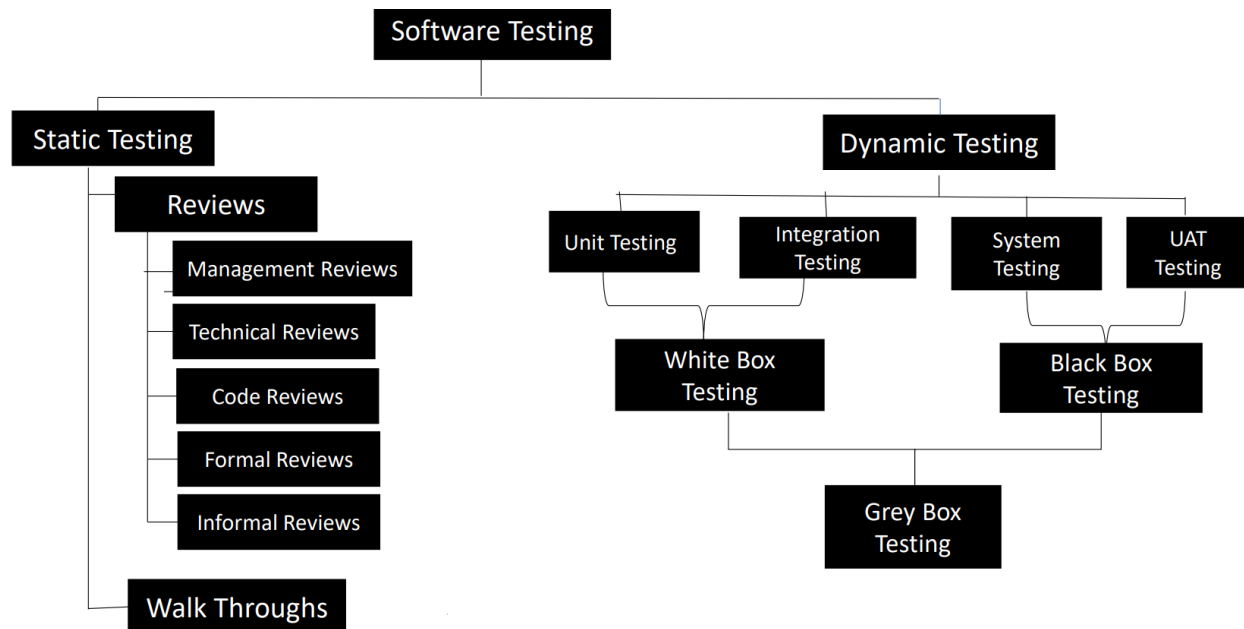
- Requirement changes are allowed in any stage of development.
- The release is speedy.
- Customers do not need to wait for a long time.
- Good communication between teams.
- It is an easy model to adopt. Flexible model.

Disadvantages:

- There is less documentation since we deliver software very fast.



Software Testing Methods and Levels



Static Testing

Static Testing is carried out by Reviews and Walk-Throughs.

1) Review:

Examining project-related work called Review.





Example: Examining requirements, design, code, etc.

Types of Reviews:

- i. **Management Review:** This review will be conducted by a high-level management team (Business Analyst, Project Manager) to monitor the project status.
- ii. **Technical Review:** This review will be conducted by the technical team to decide the best approach to design implementation.
- iii. **Code Review:** This review will be conducted by the programmer to decide the best approach for the coding part.

- iv. **Formal Review:** If a review is carried out by following systematic procedures and proper documentation, then those reviews are called formal reviews.
- v. **Informal Review:** If a review is conducted without following any procedures and documentation, then those reviews are called an Informal review.

Objectives of Static Testing: (Reviews)

-  To find defects in requirements.
-  To find defects in the design.
-  To identify the deviations in any process.
-  To provide valuable suggestions to improve the process.

Note: To detect defects from requirements, plan, design, and coding. We do static testing.

2) Walk Throughs:

A step-by-step code reviewed by experts called Walk Through.

Review can be done at any time and anyone, it is an informal review.

Dynamic Testing

Dynamic Testing is carried out by 3 methods such as

- (1) Whitebox Testing
- (2) BlackBox Testing
- (3) GreyBox Testing

Whitebox Testing:

After coding, before starting QA activities, testing conducted on source code by the programmer to ensure 100% code coverage or whether the code is working as per client requirement or not is called Whitebox Testing. It is collectively of 2 levels, such as

Unit Testing and **Integration Testing**.

Unit Testing: The smallest testable part in the source code of the application (or) every part of the code working as per requirement or not called Unit Testing, also called Module Testing (or) Component Testing.

A unit is a single component or module of software. So, conducting testing on a single component or module of software is called Unit Testing.

While conducting Unit Testing, to ensure code coverage, the programmer will follow white box testing techniques such as

- (a) Condition coverage
- (b) Loops coverage
- (c) Path coverage
- (d) Mutation coverage //They will check both valid inputs and invalid inputs to check the code coverage.

Integration Testing: Once the unit testing is completed, developers will integrate all source code units and check interactions between all modules, which is called Integration Testing. Types of Integration Testing:

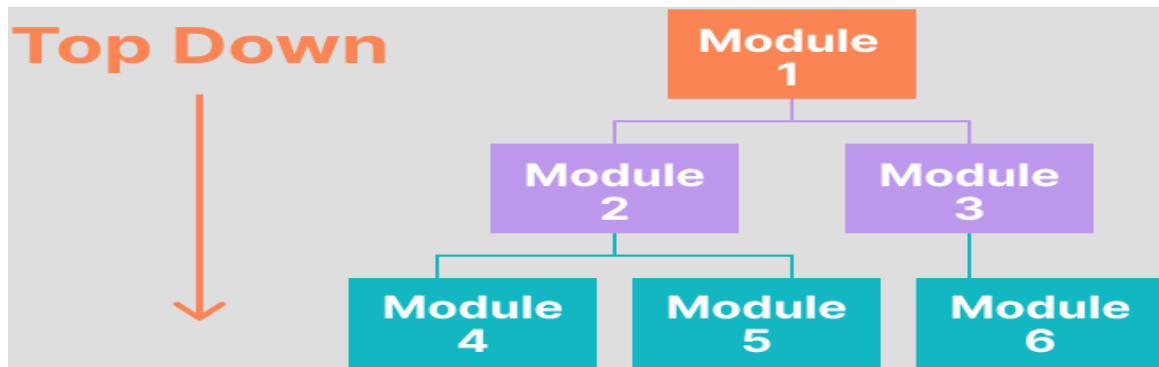
- (I) Incremental Integration Testing.
- (II) Non-Incremental Integration Testing.

Incremental Integration Testing: Incrementally adding the modules and testing the data flow between the modules one after the other, is called Incremental Integration Testing.

There are 4 approaches to Incremental Integration Testing.

- (1) Top-Down approach
- (2) Bottom-Up approach
- (3) Hybrid approach
- (4) Big bang approach

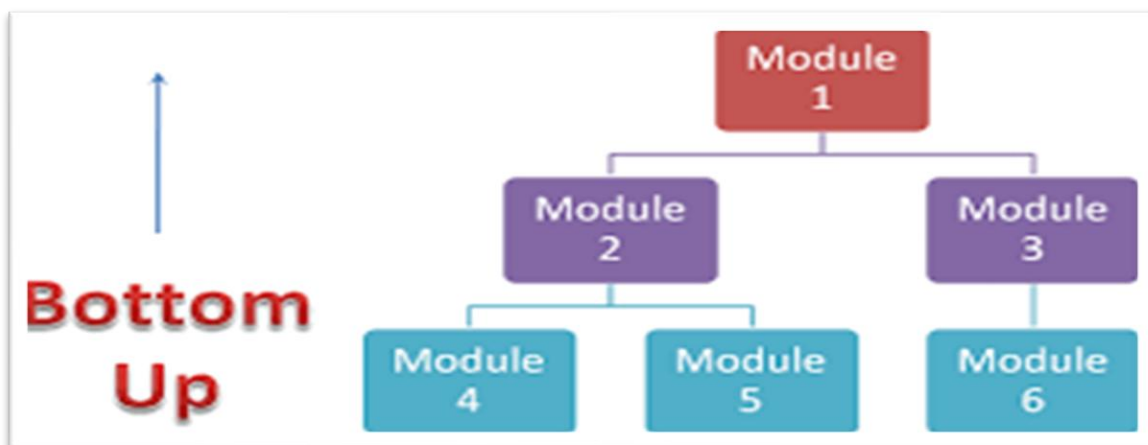
- (1) Top-Down Approach:** Incrementally adding the modules and testing the data flow between the modules. And ensures the module added is the child of the previous module. **Ex:** A module is there, and when we integrate with B module then that B module should be the child module of A.



- This approach is recommended if any incomplete programs are at the top level.
- In this approach, integration testing will be carried out from top to bottom.
- The incomplete program at the bottom level will be replaced with Stubs.

Note: Stub is a temporary code that stops user actions when the top module is incomplete.

- (2) Bottom-up Approach:** Incrementally adding the modules and testing the data flow between the modules. And ensure the module added is the parent of the previous module.



- This approach is recommended when incomplete programs are at the bottom level.
- In this, testing will be carried out from bottom to top.
- The incomplete program at the bottom level will be replaced with Drivers.

Note: Driver is a temporary code used to continue user actions when the bottom module is incomplete.

(3) Sandwich/Hybrid approach: A combination of the top-down and bottom-up approaches is the Sandwich/hybrid approach.

- In this, the middle-level modules are integrated with both Stubs and Drivers.

(4) Big Bang approach: This approach is recommended whenever 100% of the coding is completed and all source code units are available, then integration testing called as Big Bang approach.

Non-incremental Integration Testing: Here we integrate all the modules in one shot and test the data flow between the modules. (we do not prefer most of the time this type).

Drawbacks:

- We might miss the data flow between some of the modules.
- If we find any defect, we cannot understand the root cause of the defect.

Black Box Testing

After 100% coding and Whitebox testing, testing is conducted on the application by the testing engineer (or) domain experts to ensure the requirement coverage (or) to check whether the application developed as per client requirement (or) not, called BlackBox Testing, also called as Closed Box Testing or Specification Based Testing.

It is collectively of 2 levels:

- (i) System Testing
- (ii) User Acceptance Testing

System Testing

System Testing is a process of validating both Functional requirements and Non-Functional requirements.

- Validating client/business requirements is called Functional Testing.
- Validating client/business expectations is called Non-Functional Testing.

Once the project is scheduled for system testing, every testing engineer validates Functional requirements.

Whenever 100% functional requirements are validated, then the tester can go through non-functional requirements.

Types of Functional Testing:

1) Smoke Testing / Build Verification Testing

2) Sanity Testing.

3) Positive / Negative Testing

4) Re Testing

5) Regression Testing

6) Database Testing

-Data Validation, Data Integrity, Data Volume Testing

6) Exhaustive Testing

7) End-to-End Testing

8) Adhoc Testing

- Buddy Testing, Pair Testing

- Exploratory Testing, Monkey Testing.

(1) Smoke Testing / Build Verification Testing:

Functional testing starts with Smoke Testing. In this type of testing, once the build is released by the developer, check whether the application is stable or not, by checking basic functionality called Smoke Testing, also called Build Verification Testing.

(2) Sanity Testing:

Sanity testing is used to determine whether the changes or the functionality are working as expected. If the sanity testing fails, the software product is rejected by the testing team to save time and money. It is performed only after the software product has passed the smoke test and the QA team has accepted it for further testing.

(3) Positive / Negative Testing:

Positive Testing: In this type of testing, the application checks with valid inputs called Positive Testing.

Negative Testing: In this type of testing, the application checks with invalid inputs called Negative Testing.

Note:

- The objective of positive testing is to confirm customer requirements.
- The objective of negative testing is to find defects.

(4) Re Testing:

Once the defect is fixed by the developer, to check its working as per customer requirements or not, checking or testing again and again that functionality is called Re Testing.

Ex:

Destination	Duration	Start Date	Type of Holiday
 Mumbai, India(456)	6Nights / 7Days ▼	Start Date 	--Holiday Type-- ▼

Here, if “Type of Holiday” is not working, then the tester sends this defect to the developer, where the developer fixes the defect and sent back to the tester, then the tester will test the defect again and again for quality. This is called Re-Testing.

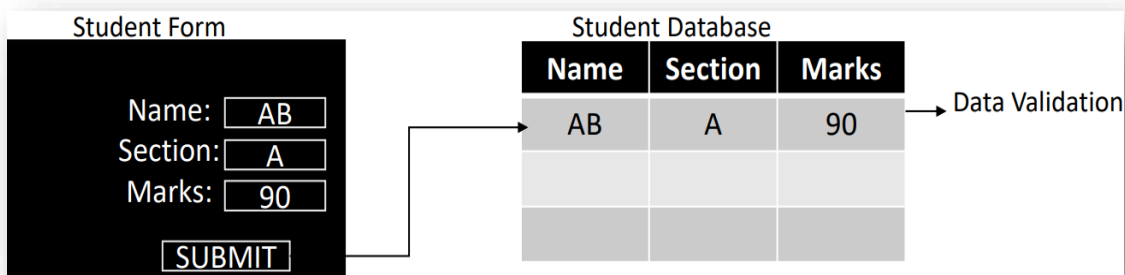
(5) Regression Testing:

Once the defect is fixed by the developer, to check the side effects of functionality along with the defect, testing defects related to all functionalities is called Regression Testing. We can ensure 100% customer satisfaction only by Regression testing, but manually it requires huge resources to perform regression testing, but it can be handled through automation tools like Selenium.

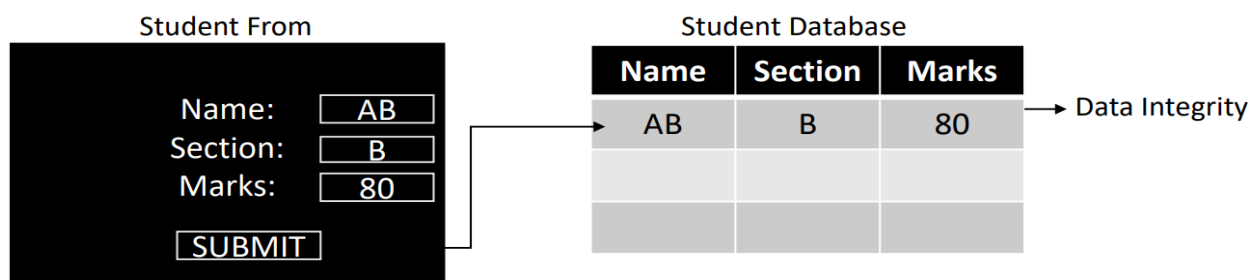
(6) Database Testing:

It is a collection of data backend application of software; every data will be inserted into the database as a table format. During database testing, we can perform the below types of testing.

- (I) **Data Validation:** In this type of testing, new data is into the database, and whether data will be correctly inserted or not, is called Data Validation.



- (II) **Data Integrity:** In this type of testing, update existing data and check correctly updated or not, called Data Integrity.



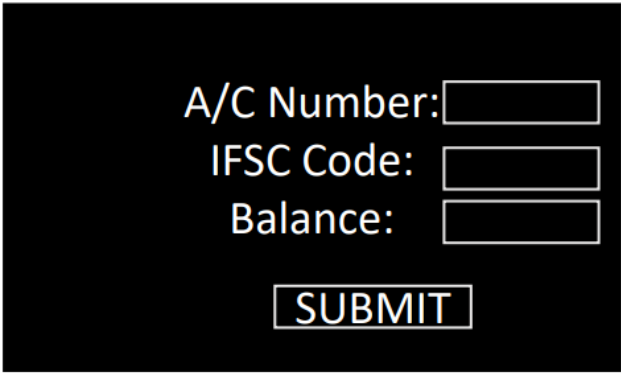
(III) Data Volume Testing: In this type of testing, checking the capacity of the database by entering sample data into the database, is called Data Volume Testing or Data Memory Testing.

(7) Exhaustive Testing:

If we test a functionality with all possible inputs called Exhaustive Testing. We can perform Exhaustive testing on Account Numbers, and IFSC code functionalities, but to perform exhaustive testing on maximum balance requires huge resources. So, it requires Automation help.

Ex:

Bank Application



A/C Number:

IFSC Code:

Balance:

(8) End-To-End Testing:

If we conduct testing on an application from start to end based on all requirements of an application, it is called End-To-End Testing.

We can perform this testing whenever we have sufficient time.

(9) Adhoc Testing:

Due to a lack of time, instead of testing all modules, testing the main modules of an application called Adhoc Testing. Due to lack of time, under Adhoc testing, testers will perform the below types of testing.

- Buddy Testing, Pair Testing
- Exploratory Testing, Monkey Testing.

Buddy Testing: Due to a lack of time, the testing team can join with the development team to continue development and testing parallelly to complete it as soon as possible, a process called Buddy Testing.

Pair Testing: Due to a lack of time, the testing team will combine as a pair or group and perform testing on the project. While working, they can share knowledge and ideas about a project called Pair Testing.

Exploratory Testing: Due to a lack of skills, exploring knowledge on a project by using search engines, interacting with seniors, or verifying project-related documents, then working on the project is called Exploratory Testing.

Monkey Testing: Due to lack of time, without documents, processes, or standards work on the project randomly to break the system is called Monkey Testing.

Non-Functional Testing

Once the project is validated by all functional requirements to ensure customer expectations, the testing engineer can perform the below types of non-functional testing.

- (1) GUI Testing
- (2) Usability Testing
- (3) Compatibility Testing
- (4) Security Testing
- (5) Installation Testing
- (6) Recovery Testing
- (7) Standard Testing
- (8) Performance Testing

(1) GUI Testing: Graphical User Interface Testing or GUI testing is a process of testing the user interface (UI) of an application.

A GUI includes all the elements such as menus, checkboxes, buttons, colors, fonts, sizes, icons, content, and images.

GUI testing checklist:

- Testing the size, position, width, height of the elements.
- Testing of the error messages that are getting displayed.
- Testing the different sections of the screen.
- Testing of the font whether it is readable or not.
- Testing of the screen in different resolutions with the help of zooming in and zooming out.
- Testing the alignment of the texts and other elements like icons, buttons, etc. are in proper place or not.
- Testing the colors of the fonts.
- Testing whether the image has good clarity or not.
- Testing the alignment of the images.
- Testing of the spelling.
- The user must not get frustrated while using the system interface.
- Testing whether the interface is attractive or not.
- Testing of the scrollbars according to the size of the page if any.
- Testing of the disabled fields if any.
- Testing of the size of the images.
- Testing of the headings whether it is properly aligned or not.
- Testing of the color of the hyperlink.
- Testing UI Elements like button, textbox, text area, check box, radio buttons, drop downs ,links etc.

(2) Usability Testing: In this type of testing from the customer's point of view, whether the application is user-friendly or not means the user can understand and operate the application easily or not.

During this testing, validates application provided context-sensitive help or not to the user.

(3) Compatibility Testing (Configuration Testing):

In this type of testing, to check the application compatibilities, and validate both software compatibilities and hardware compatibilities.

Software Compatibility: In this testing application validated by various operating systems, various browsers supporting or not called Software Compatibility.

Hardware Compatibility: In this testing application validated by various hardware device configurations supporting or not called Hardware Compatibility.

(4) Security Testing: To check whether the application is secure or not, the security tester will perform below types of testing.

Authentication Testing: In this type of testing, we need to check whether the valid user is accepted or not and whether the invalid user is rejected or not, called Authentication Testing.

Authorization Testing (Access Control): In this testing, whenever the login as a level of user into the application, check whether he crosses any beyond user limits or not, called Authorization Testing or User Permission Testing.

Encryption Testing: In this testing, to check whether the securable data default converted into encrypted format or not, called Encryption Testing.

(5) Installation Testing: In this testing, we need to follow installation guidelines which are defined under the installation document. During this test, we must verify the following steps:

- While installing, the application is user-friendly or not?
- After installation, check memory.
- Verify the uninstallation process.

(6) Recovery Testing: This type of testing, checks how the system handles unexpected events such as power failure, application crash, etc.

(7) Standard Testing: In this type of testing, we need to check whether the application developed by following company standards or not such as ISO [International Standard Organization] etc.

(8) Performance Testing: Speed in processing to check the performance of the application, testers will follow the below types of testing. JMeter tool

(I) Load Testing: The execution of software under the customer's expected configuration and customer expected load, to estimate the speed in processing and whether is there any chances of breaking the application, called Load Testing.

Note: gradually (slowly) increase the load on the application.

(II) Stress Testing: The execution of software customer expected configuration and more than the customer expected load, called Stress Testing. Note: suddenly increase/decrease the load and check the speed and stability of the application.

(III) Volume Testing: Volume means size, testing how much data the application can handle.

User Acceptance Testing

User Acceptance Testing is performed after the product is thoroughly tested. In this testing, the client uses the application to make sure everything is working as per the requirements and perfectly in the real-world scenario.

User Acceptance Testing is also known as End User Testing.

User Acceptance Testing is conducted at 2 levels: (1) Alpha Testing and (2) Beta Testing.

(1) Alpha Testing: Alpha Testing is done by users or customers in a development or testing environment means they will come back to the company where ever software is developed and do the testing and this testing is done before the software is released for Beta test.

(2) Beta Testing: Beta Testing is done by users or customers in a real/customer environment.

Note: After Alpha and Beta testing, the product will be released to the production environment and then actual users will start using the software.

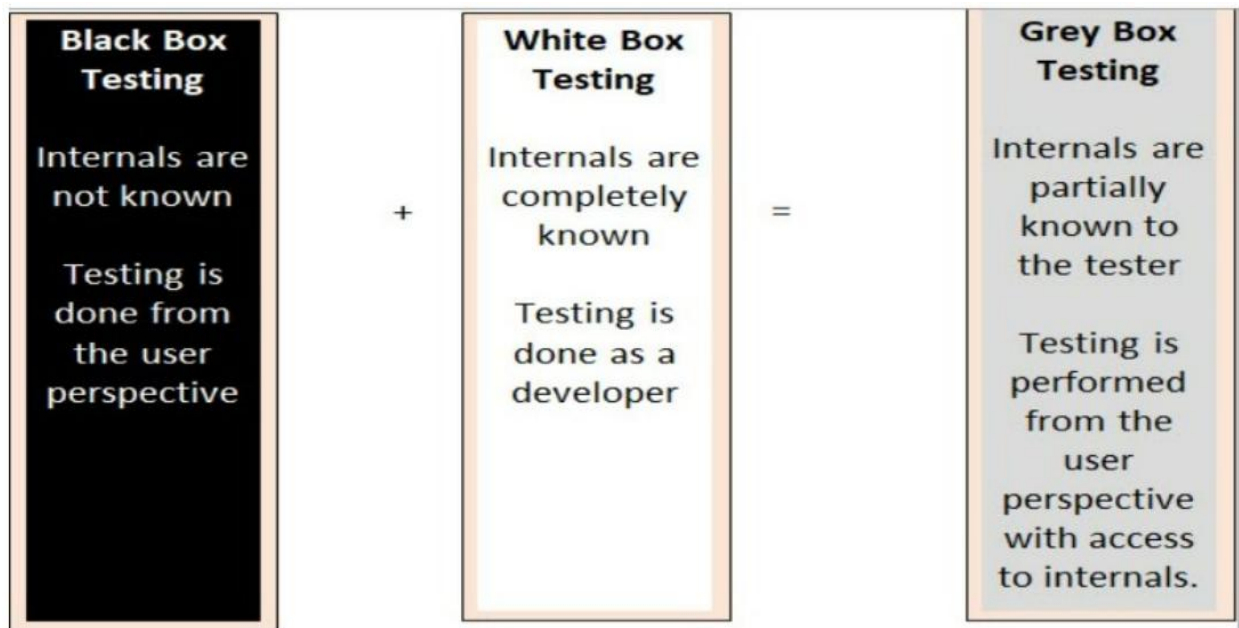
Alpha Testing Vs Beta Testing:

Alpha Testing	Beta Testing
First phase of testing in customer validation.	Second phase of testing in customer validation.
Performed at developer's site – testing environment. Hence the activities can be controlled.	Performed in real environment, and hence activities cannot be controlled.
Whitebox and/or Blackbox testing techniques are involved.	Only Blackbox testing techniques are involved.
Build released for Alpha Testing is called Alpha Release.	Build released for Beta Testing is called Beta Release.
System Testing is performed before Alpha Testing.	Alpha Testing is performed before Beta Testing.
Issues/Bugs are logged into the identified tool directly and are fixed by developer at high priority.	Issues/Bugs are collected from real users in the form of suggestions/feedbacks and are considered as improvements for future releases.
Alpha Testing is used to evaluate the quality of the product.	Beta Testing is used to evaluate customer satisfaction.
It focus on finding bugs.	It focus on collecting suggestions/feedback and evaluate them effectively.

Grey-Box Testing

Grey-Box Testing It is also called as Grey-Box Analysis. In this testing, the tester has limited knowledge of internal details of the design than test application called Grey-Box Testing.

- This testing technique is a combination of Black box testing and White box testing.
- In Black box testing, the tester does not have any knowledge about the code. They have information for what will be the output for the given input. In White box testing, the developer has complete knowledge of the code.
- Grey box tester knows the code, but not completely.



Purpose

- The purpose of grey box testing is to improve the quality of the product for which it covers both functional and non-functional testing altogether, which saves time and the lengthy process to test the application.
- Another purpose is to have the application tested from the user's perspective rather than the designer's opinion, and to get the developers enough free time to fix the bugs.

Examples

- While testing a website, if the tester clicks on any link and it comes up with an error, the grey box tester can make changes in the HTML code to check the same.
- In this scenario white box testing is being done by changing the code in HTML directly and can be verified in real-time. and as the tester is testing the changes at the front end–black box testing is also being done. Combination of the White box and Black box results in grey box testing.

Advantages: Grey box testing has several advantages. A few of them are mentioned below:

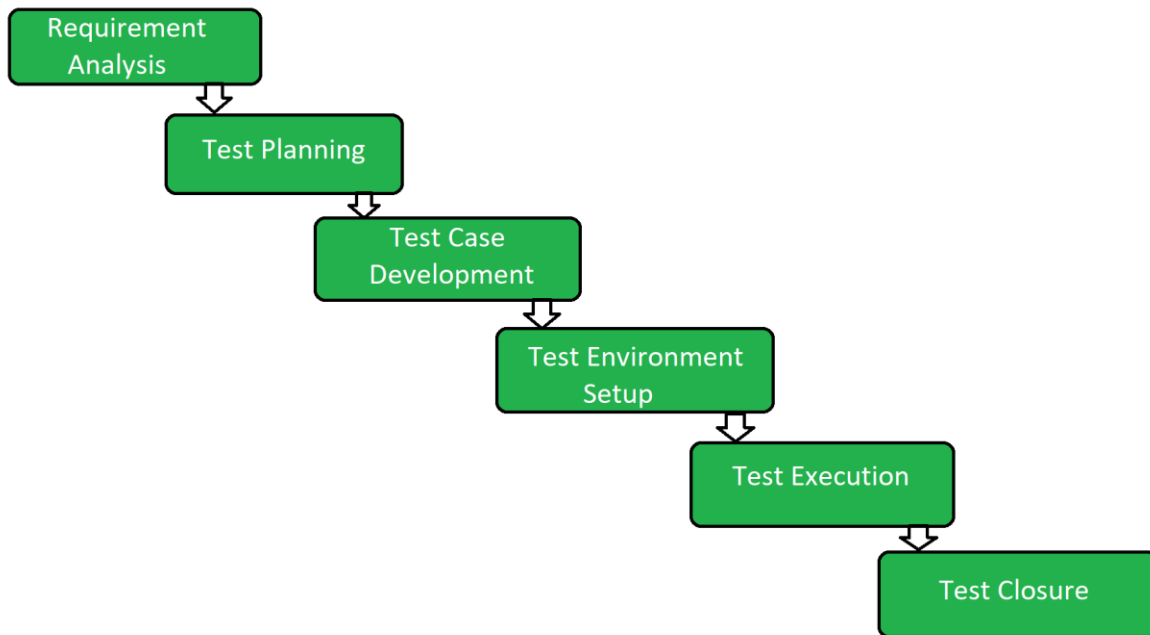
- Improves the quality of the product.
- Complex applications can be tested with ease.
- Do not require the tester to have complete knowledge of coding.
- Obtain benefits of both black box and white box testing.

Difference between Black Box, White Box and Grey Box Testing

Black Box Testing	White Box Testing	Grey Box Testing
Testers does not have any knowledge and information for the internal structure.	The internal structure is completely known.	The internal structure is partially known.
Also known as Closed box, Functional testing and Data-driven testing.	Also known as Glass, Clear box, Structural testing, or Code-based testing.	Also known as Gray box testing or translucent testing.
Done by testers, developers and the end-users.	Done by testers and developers only.	Done by testers, developers and the end-users.
Testers has nothing to do with the internal working of the system.	Testers has no knowledge of the internal working of the system.	Testers has partial knowledge of the internal working of the system.
Hidden errors cannot be found easily.	Hidden errors can be found easily as internal working is well to the developer.	Not easily found but can be found at user level.

STLC

STLC stands for “Software Testing Life Cycle,” it will explain all the implementation activities of the testing process.



1) Test Analysis: In this phase, to understand project requirements, the testing engineer starts analyzing various requirement documents such as

- (I) FRS and
- (II) UseCase documents.

While analyzing requirements, if we need any clarifications, we update the RCN (Requirement Clarification Note) and send it to the BA (Business Analyst) (Domain Experts).

2) Test Planning: At this phase prepare 2 documents such as

- (I) Test Strategy and
- (II) Test Plan documents.

- Test Strategy is a high-level plan or project-level plan that Test Managers prepare.
- Based on the strategy, the number of Test Plan documents will be prepared by the Test Lead for the Responsible module.

- 3) Test Design:** In this phase, design the requirements as 2 documents such as
- (I) Test Scenarios and
 - (II) Test Cases
- What to test (or) one unique test condition is called Test Scenario.
 - How to test and step-by-step process is called Test Cases. The number of test scenarios and test cases updates into RTM (Requirement Traceability Matrix).
- 4) Test Environment:** It is a combination of software and hardware configurations such as browsers, operating systems, system configurations, etc.
- 5) Test Execution:** Whenever test cases are prepared by testing engineers and as per the release date, the build is released by the developer to compare expected values and actual values, and every test case executes on the build.
- 6) Defect Reporting:** During execution, if any mismatches between expected and actual called a Defect. With the help of the defect tracking tool, the defect will be reported to the developer.
- 7) Test Closer:** Whenever all the test cases are executed successfully and all major defects are fixed, then the tester can sign off from the current project called Test Closer.

Test Planning

Once the project is scheduled for system testing, prepare 2 types of plan documents, such as

- (1) Test Strategy and
- (2) Test Plan document.

(1) Test Strategy Document:

- It is a high-level plan document developed by the Test Manager with the help of the Project manager, based on BRS and SRS/FRS documents.
- This document defines a software testing approach to achieve testing objectives.
- It is a static document, that is not updated. Some companies include a Test Strategy with the plan and prepare only one Test Plan document.

(2) Test Plan Document:

- A test Plan document is a module-level plan prepared by the Test Lead based on the Test Strategy and requirement documents, following the IEEE 829 format.

- IEEE stands for Institute of Electrical and Electronic Engineers.
- This plan document contains Scope, Resources, Roles & Responsibilities, etc.
- Test Plan will explain what to test (or) when to test requirements.

Review Test Plan Document

Once the Test Plan is prepared by the Test Lead before start analyzed by Team members, to check whether the plan is prepared as per the client's requirement or not perform the below reviews –

- (I) **Self Review:** Reviewed by the same person.
- (II) **Peer Review:** Reviewed by colleagues.
- (III) **Lead Review:** Reviewed by Team Lead.
- (IV) **Manager Review:** Reviewed by Test Manager or Project Manager.
- (V) **Client Review:** Reviewed by the client.

Test Analysis

Once the test plan is prepared and reviewed, in this phase, the testing engineer will analyze various requirements such as SRS/FRS, and use case documents to understand what to be tested and how to test all requirements.

While analyzing requirements, if there are any questions or doubts, we record out in a process template called **RCN [Requirement Clarification Note]** and we send this document to BA to get clarifications.

Business Requirement Specification [BRS] Document Template:

1.0	Introduction
	1.1 Client Introduction
	1.2 Project Introduction
2.0	Existing System
3.0	Drawbacks in existing system
4.0	Proposed system
5.0	System Architecture
6.0	Business Requirements

System Requirement Specification [SRS] Template:

SRS is also called as FRS [Functional Requirements Specification] document (or) FD [Functional Document] (or) BRD [Business Requirement Document] (or) BDD [Business Design Document].

1.0	Overview
2.0	Prototype
3.0	Form/Page Elements
4.0	Business Validation or input validation and error states
5.0	Use case diagram/DFD's/Task flow diagram
6.0	Use case

RCN Template:

Project Name	Gmail				
Module Name	Login				
Prepared By					
Prepared Date					
#	Requirement Specification Reference	Clarification Required	Clarification Provided	Clarification Provided By	Clarification Provided Date
01 2022	FRS/Usecase	Forget Password			

Test Design

After understanding various requirement documents, those requirements will be designed as 2 documents, such as (a) Test Scenario and (b) Test cases

- What to test (or) one unique test condition (or) one line order of requirement is called a Test Scenario, which defines the High-Level Test Design of the requirement.
- How to test (or) step-by-step process is called Test case, which defines Low-Level Test Design of the requirement.

Example 4: Test Scenario for washing machine:

- (1) Verify capacity
- (2) Verify model (top load or front load)
- (3) Verify functionality
- (4) Verify power supply
- (5) Verify speed
- (6) Verify design and color
- (7) Verify whether it is automatic or not
- (8) Verify dryer

IEEE829 Test Case Format

A	B	C	D	E	F	G	H	I	J	K
TestCase Template										
	Project Name	Gmail	Reviewed By							
	Module	Login	Reviewed Date							
	Document References	FRS	Approved By							
	Author		Approved Date							
	Created Date									
TestCaseID/Name	Testcase_Description	Pre_Condition	StepNo	Step Description	Input Fields	Expected Result	Actual result	Testcase Result	Testcase Priority	Comments

Project Name: Name of the current project

Module Name: Name of responsible module

References: Name of requirement documents

FRS/UseCase Author: Name of Testing Engineer Created

Date: Date of test case designing

Testcase ID/Name: One unique ID or name to identify a document in the future.

Testcase Description: Name of Test Scenario

Precondition: List of steps that need to be completed before requirement

Step no: Serial number

Step Description: How to test, step by step clear requirement

Input Fields: Prep is input data by using test case design techniques

Expected Result: Client expectation from the requirement document

Actual Result: After execution result is updated as an actual result

Testcase Result: If the expected is equal to the actual value, then the result should be passed, else fails.

Testcase Priority: It is used to define the importance of the test case

Comment: Comments if any.

Reviewed By, Reviewed Date: After the test case design is reviewed by the Test Lead the date is also updated by the Test Lead.

Approved By, Approved Date: Cases should be approved by the Test Manager or Project Manager

Black Box Design Techniques/Test Data (or) Input Techniques

Test Data/Design technique used to prepare data for testing which can cover each and everything of the functionality.

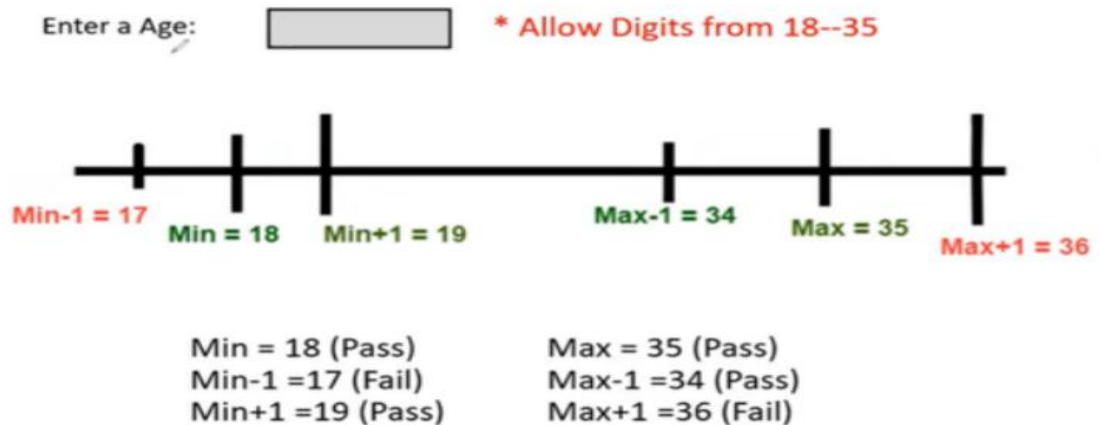
There are 5 types of Test Data or Test Design or case design techniques:

- 1) Boundary Value Analysis (BVA)
- 2) Equivalence Class Partition (ECP)
- 3) Decision Table (DT)
- 4) State Transition (ST)
- 5) Error Guessing (EG)

1. Boundary Value Analysis (BVA):

- It defines the size and range of the inputs.
- While using the BVA technique, the starting value is considered as a Lower Boundary, and the ending value is considered as an upper Boundary.
- Instead of checking all possibilities, lower boundary checks with +1 and -1, and ending value checks with -1 and +1.

Example:

BVA:**2. Equivalence Class Partition (ECP):**

- It defines the type of inputs/values such as alphabets, numeric, blank spaces, special characters, etc.
- In ECP, first, we must divide what is valid and what is invalid, then divide into groups.
- But while making groups, instead of checking all possibilities, starting, middle, and ending values will be considered.

Example:

Enter a Number: <input type="text"/> * Allow Digits from 1--500		
Normal Test Data	Divide values into Equivalence Classes	Test Data using ECP
1	-100 to 0 → -50 (Invalid)	-50
2	1 - 100 → 30 (Valid)	30
3	101 - 200 → 160 (Valid)	160
4	201 - 300 → 250 (Valid)	250
.	301 - 400 → 320 (Valid)	320
.	401 - 500 → 450 (Valid)	450
.	501 - 600 → 550 (Invalid)	550
500		

Example 2:

Name: <input type="text"/> * Allow only alphabets	
Divide values into Equivalence Classes	Test Data using ECP
A..Z → (Valid)	XYZ
a..z → (Valid)	zyz
Special Characters → (Invalid)	@#\$%
Spaces → 250 (Invalid)	Xy_z
Numbers → 320 (Invalid)	1234

3. Decision Table (DT)

- This is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form.

CONDITIONS	EMAIL	V	V	I	I	B	V	I	B	B
	PASSWORD	V	I	V	I	V	B	B	I	B
ACTIONS	EXPECTED RESULT	PASSED	FAILED	FAILED	FAILED	FAILED	FAILED	FAILED	FAILED	FAILED

Example 1:

Verify Radio button:

Value	Expected
Select one way	one way should be active, Round trip and Multi city should be inactive.
Select Round trip	Round trip should be active, One way and Multi city should be inactive.
Select Multicity	Multicity should be active, One way and Round trip should be inactive

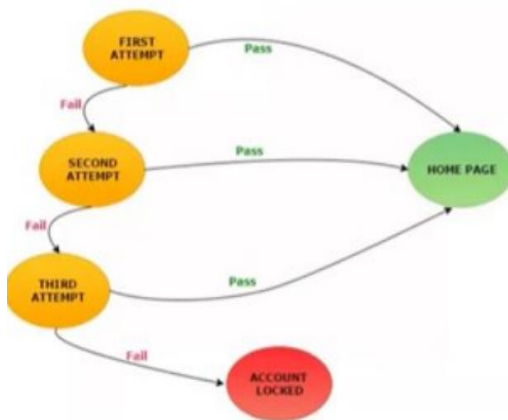
Example 2:

Decision Table for Upload Screen								
You can upload only '.jpg' format image file size less than 32kb resolution 137*177.								
<div> <input type="text" value="upload photo"/> *upload .jpg file with size not more than 32kb and resolution 137*177 </div> <div> <input type="button" value="upload"/> </div>								
Conditions	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8
Format	.jpg	.jpg	.jpg	.jpg	Not .jpg	Not .jpg	Not .jpg	Not .jpg
Size	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb	Less than 32kb	Less than 32kb	>= 32kb	>= 32kb
resolution	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177	137*177	Not 137*177
Output	Photo uploaded	Error message resolution mismatch	Error message size mismatch	Error message size and resolution mismatch	Error message for format mismatch	Error message format and resolution mismatch	Error message for format and size mismatch	Error message for format, size, and resolution mismatch

4. State Transition:

- Every application has different states (user interface), the state will navigate from one state to another state based on user operations.
- In this technique, we provide positive as well as negative input test values to evaluate the system behavior.

- Take an example of login page of an application which locks the user name after three wrong attempts of password.



STATE	LOGIN	CURRENT PASSWORD	INCORRECT PASSWORD
S1	First Attempt	S4	S2
S2	Second Attempt	S4	S3
S3	Third Attempt	S4	S5
S4	Home Page		
S5	Display a message as "Account Locked, please consult Administrator"		

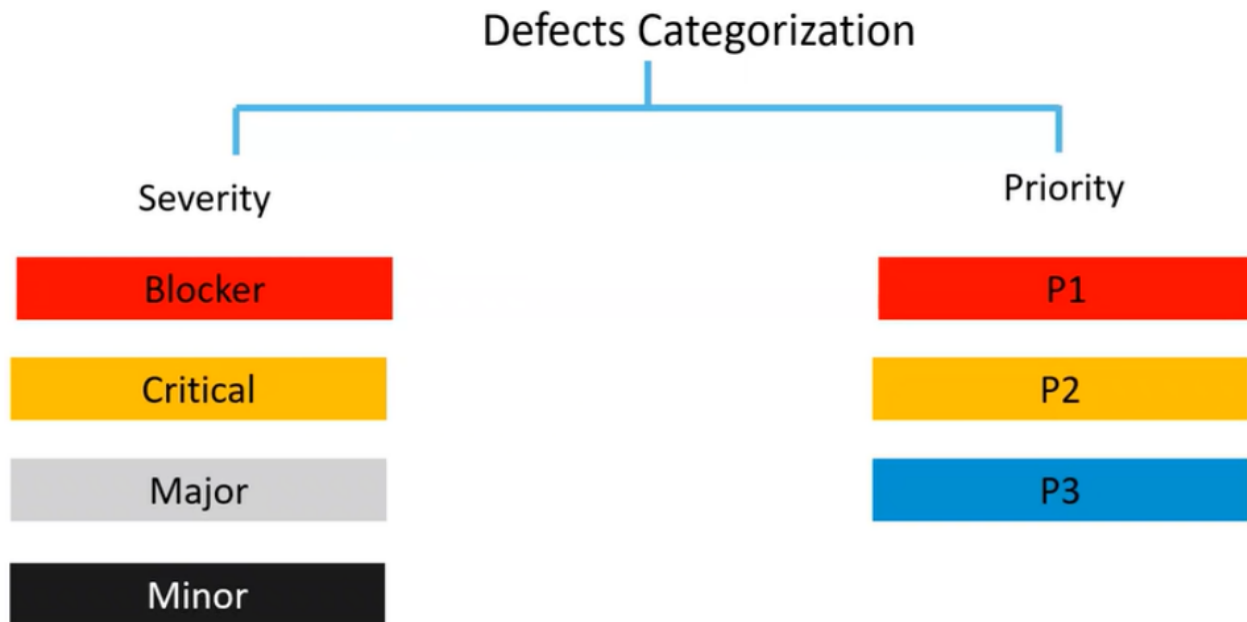
5. Error Guessing:

- Error guessing is one of the testing techniques used to find bugs in a software application based on a tester's prior experience.
- In Error Guessing we do not follow any specific rules.
- It depends on the tester's analytical skills and experience.

Examples are: -

- Submitting forms without entering values.
- Entering invalid values in the fields etc.

Priority and Severity



Defect Severity

- Severity describes the seriousness of defect and how much impact on Business workflow.
- **Defect severity can be categorized into four class**
 - **Blocker(Show stopper):** This defect indicates nothing can proceed further.
 - Ex: Application crashed, Login Not worked
 - **Critical :** The main/basic functionality is not working. Customer business workflow is broken. They cannot proceed further.
 - Ex1: Fund transfer is not working in net banking
 - Ex2: Ordering product in ecommerce application is not working.
 - **Major:** It cause some undesirable behavior, but the feature/application is still functional.
 - Ex1: After sending email there is no confirm message
 - Ex2: After booking cab there is no confirmation.
 - **Minor:** It won't cause any major break-down of the system
 - Ex: Look and feel issues, spellings, alignments.

Source: ISTQB

Priority: How soon must we fix the defect? Urgency to repair the defect.

Defect Priority

- Priority describes the importance of defect.
- Defect Priority states the order in which a defect should be fixed.
- **Defect priority can be categorized into three class**
 - **P0 (High)** : The defect must be resolved immediately as it affects the system severely and cannot be used until it is fixed.
 - **P1 (Medium)**: It can wait until a new versions/builds is created
 - **P2 (Low)**: Developer can fix it in later releases.

Note: - Severity and Priority will be provided by Testers, but Priority can be changed by the Developer, BA also, but Severity cannot be changed, it should be provided by Testers only.

High severity, priority and low severity, priority defects

		Priority	
		High	Low
Severity	High	Login is taking to the blank page.	About Us link is going to blank page.
	Low	After user is logged into application, he can see Home Page. But there is spelling mistake in <u>Home Page</u> .	User opened contact page. Email ID has spelling mistake.

More examples:

- **Low priority-Low severity** - A spelling mistake in a page not frequently navigated by users.
- **Low priority-High severity** - Application crashing in some very corner case.
- **High priority-Low severity** - Slight change in logo color or spelling mistake in company name.
- **High priority-High severity** - Issue with login functionality.(user is not able to login to the application)
- **High Severity- Low Priority** - Web page not found when user clicks on a link (user does not visit that page generally)
- **Low Priority- Low Severity** - Any cosmetic or spelling issues which is within a paragraph or in the page

Defect Reporting**Defects/ Bugs:**

- Any mismatched functionality found in an application is called a Defect/Bug/issue.
- During Test execution, test engineers report mismatches as defects.
- Defect reporting tools:
 - Clear Quest
 - DevTrack
 - Jira
 - Quality Center
 - Bug Jilla, etc.

IEEE829 Defect Report Template:

Project Name

Defect_Id	Defect Description	Reproducible [Y/N]	Reproductive steps	Defect Severity	Defect Priority	Defect Status New/Reopen	Detected By	Detected Date	Detected in Version	Fixed By

Project Name: Name of the current project**Defect_Id:** One unique ID or name to identify a document in the future.

Defect Description: Prepare a step-by-step description of the defect.

Reproducible [Y/N]: While executing test cases if the defect producing every time, update the status as “Yes” or while executing defect producing only sometimes, then update the status as “No.”

Reproducible Steps: If reproducible is “Yes”, then attach test case If reproducible is “No”, then attach test case and screenshot.

Defect Severity: It defines the impact of the defect as Blocker, Critical, Major, and Minor severities.

Defect Priority: It indicates urgency to fix the defect as a High, Medium, or Low.

Defect Status: If a defect occurs as first time, then update the status as “New.” Once the defect is fixed also, the same defect is repeated, and then the update status is Reopen.

Detected By: Name of the testing engineer.

Detected Date: Date of defect identified.

Detected in Version: Name/Number of build versions - Fixed By/Fixed Date: Both will be updated by a developer after defect fixing.

Date of Closure: Once the defect is fixed by the developer, during regression testing, if the defect is fixed as per requirement, then update the date of closure or update the status as reopen by the testing engineer.

Bug Life Cycle

Once defects are identified by the Tester, those defects will be documented in a Defect Report Template or Bug Tracking Tool, with the status “New.”

Once all new defects are reported to the developer, the developer will analyze them

- If a reported defect is invalid, update the status as “Rejected.”
- If the reported defect is valid, then based on priority, the developer updates the status as “Open.”
- If the developer needs clarification about the defect, he updates the status as “Hold.”
- If a defect is postponed to the next phase, then the status will be “Differed.”
- If the defect is already reported, then the status will be updated as “Duplicate.”

Once this analysis is completed, the developer starts fixing defects. Once defects are fixed, the developer updates the status as Fixed or Resolved.

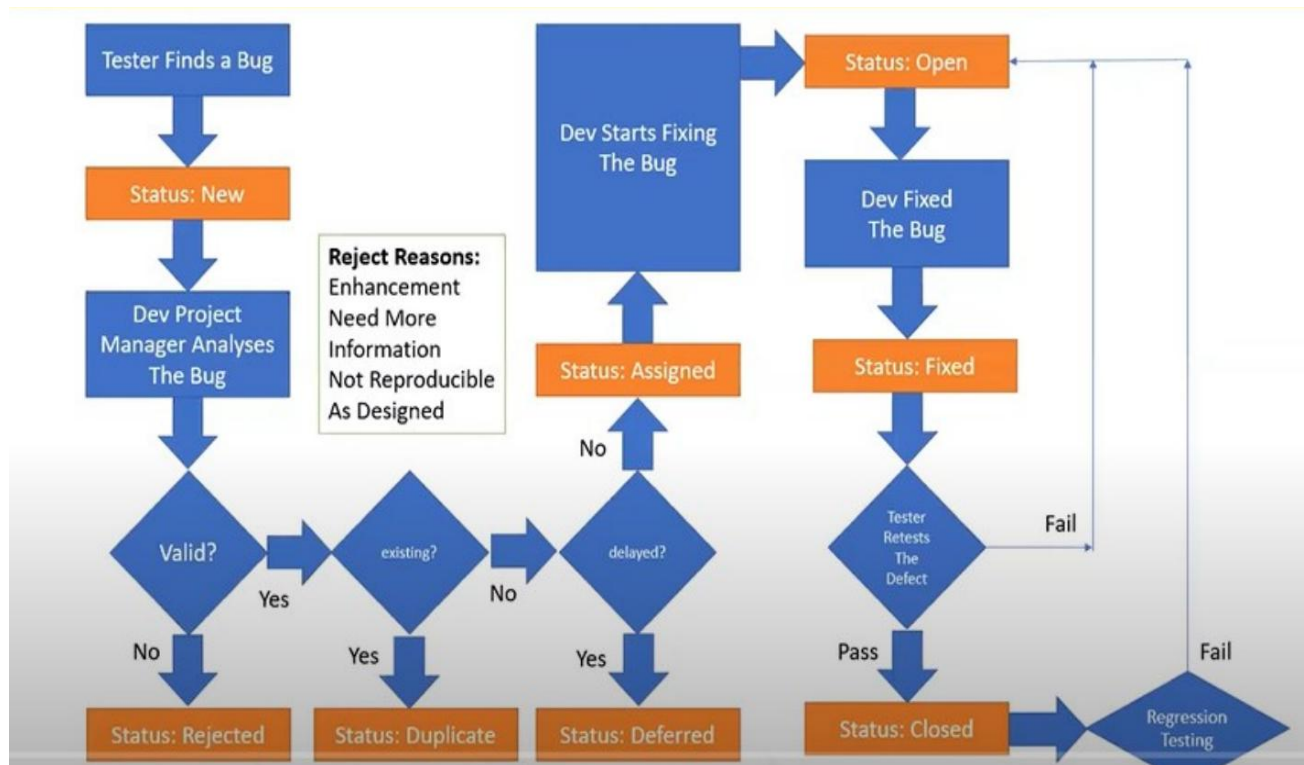
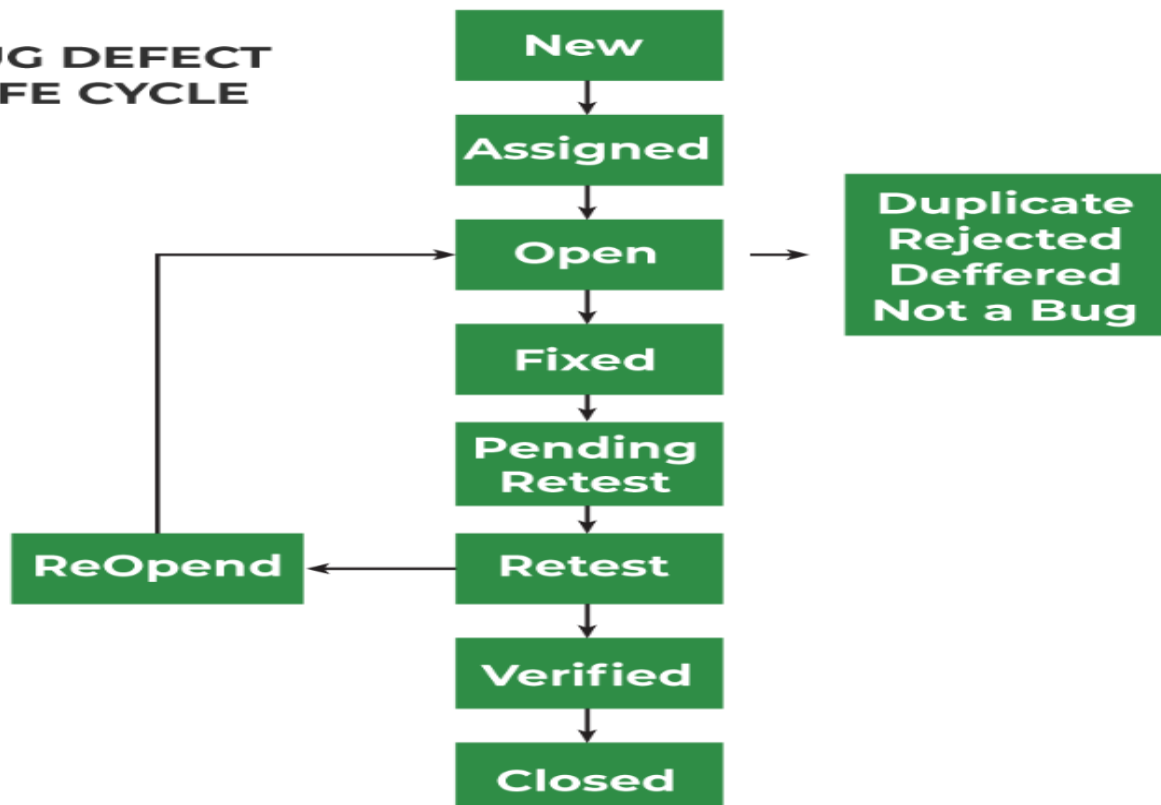
Once all defects are fixed, before release to the testing engineer, modify the build version and update as a modified build.

The tester will start Re/Regression testing on the build to confirm whether the defect is fixed as per the client's requirement or not.

If the defect is fixed in Regression testing, update the status as “Close,” or else the status will be updated as “Re-open.”

If any new defects are identified in the modified build, the same will be documented with the status “New,” and the cycle will continue till all defects are closed.

Note: suppose the reported bug is fixed and closed in one build and then in upcoming builds this defect is detected again, then we should raise a new bug by specifying a new build number, but we cannot reopen the closed bug.

BUG LIFE CYCLE DIAGRAM**BUG DEFECT LIFE CYCLE**

Test Closer / Signoff

Whenever all test cases are executed successfully and all major defects are fixed, before the signoff current project, the Test Lead will perform final Regression testing on doubtful functionalities.

During this testing, if any defect is identified then it is called a “Golden Defect.” Those defects as soon as possible reported to the developer during signoff from the project. The test Lead will prepare a test summary document. Final Test Summary Documents:

- Test summary document
- Testcase summary document
- Test execution summary document
- Defect summary document.

Principles of Software Testing

1. Start software testing at early stages. This means from the beginning when you get the requirements.
2. Test the software to find the defects.
3. Highly impossible to give bug-free software to the customer.
4. Should not do Exhaustive testing. This means we should not use the same type of data for testing every time.
5. Testing is context-based. This means deciding what types of testing should be conducted based on the type of application.
6. We should follow the concept of the Pesticide Paradox. This means, that if you are executing the same cases for a longer run, they will not find any defects. We must keep updating test cases in cycle/release to find more defects.

7. We should follow defect clustering. This means some of the modules contain most of the defects. the modules. we can identify such risky modules. 80% of the problems are found in 20% of the modules.

8. Absence of Error Fallacy: If we find more defects without fulfilling user needs, then those defects are not helpful.