# Testing and Maintenance

About Software Testing

- Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free.

- The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

- It is a process of identifying the correctness of software by considering its all-attributes Reliability, Scalability, Portability. Re-usability & Usability of the product.

- Testing is mandatory because it will be a dangerous situation if the software fails any of time due to lack of testing. So, without testing software cannot be deployed to the end user.

✓Software Testing is done at every phase level in software development life cycle(SDLC).

✓Performed by Software Tester, Software Developer, Project Manager, End-User.
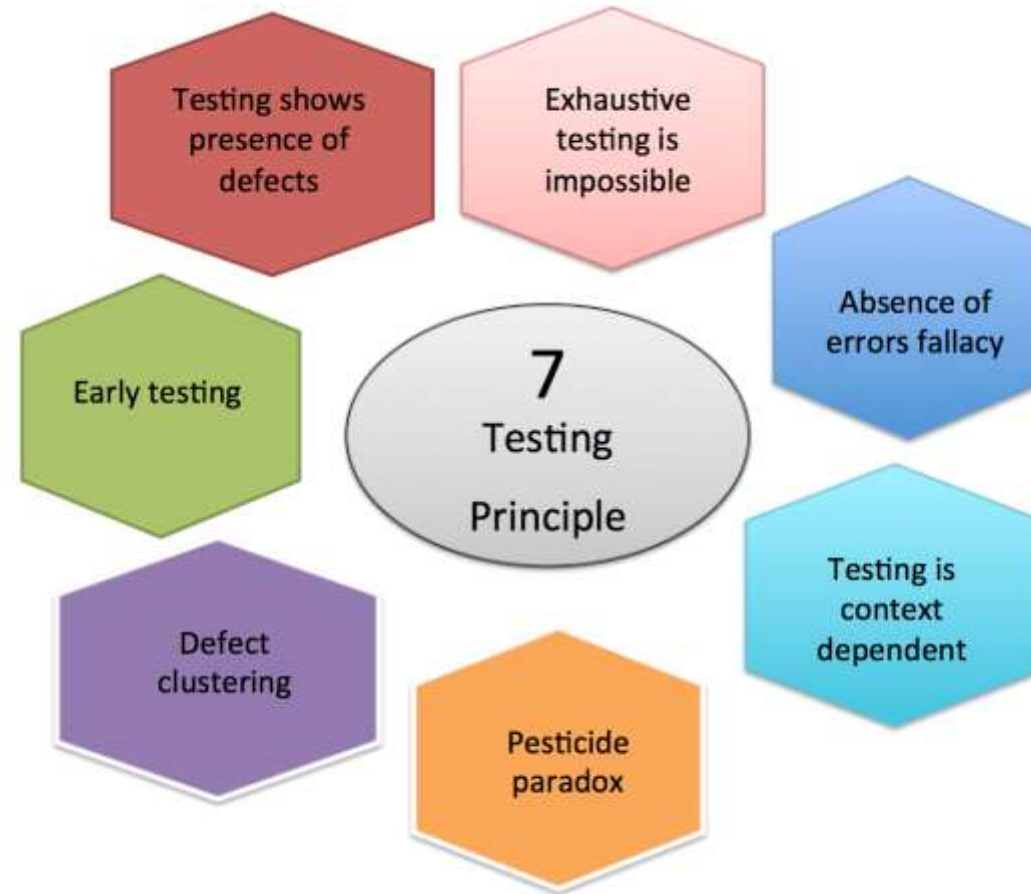
Types of Software Testing
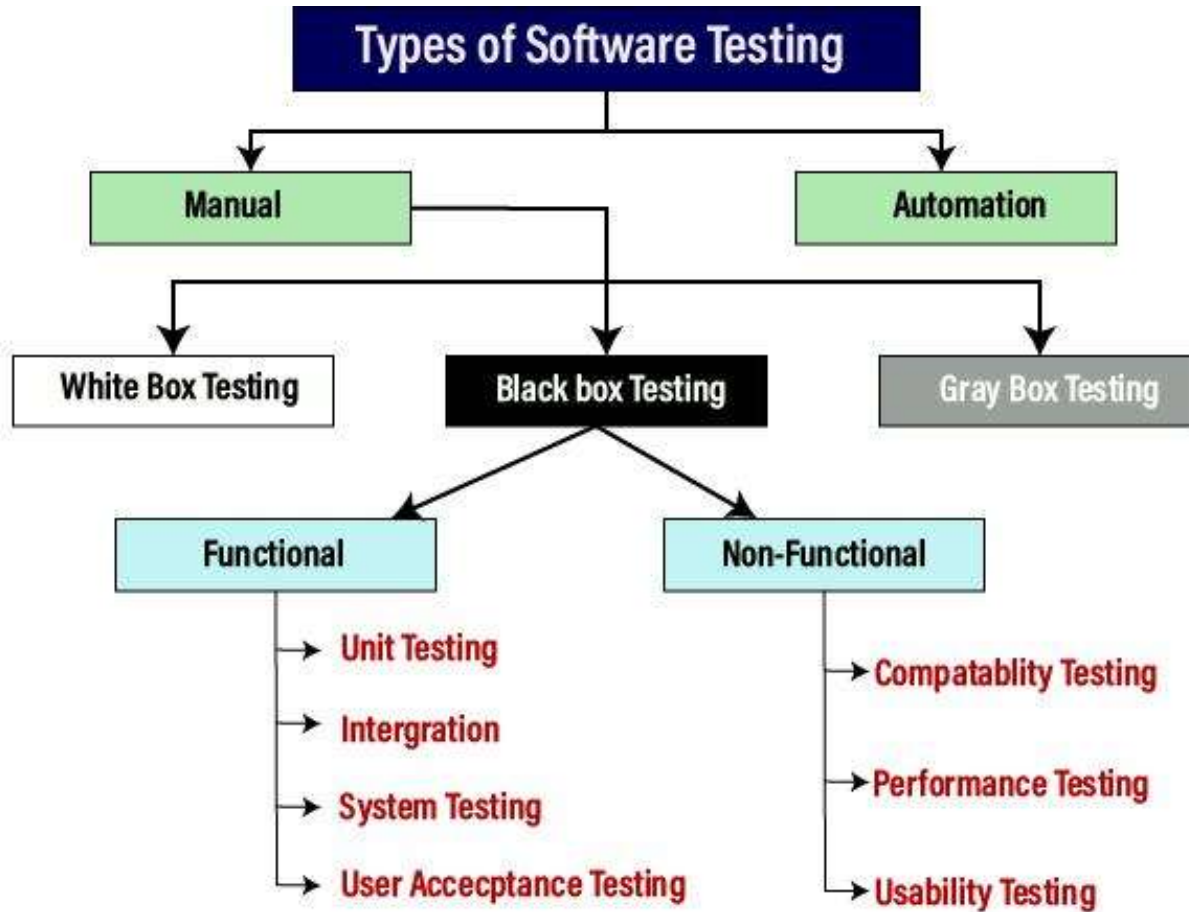
What is Test Cases?

- The test case includes set of actions or conditions, using which a testing engineer can compare expected result and actual results.

- They determine whether a software product is functioning as per the requirements of the customer.

Test Scenario ID	Login-1		Test Case ID	Login-1A	
Test Case Description	Login – Positive test case		Test Priority	High	
Pre-Requisite	A valid user account		Post-Requisite	NA	

Test Execution Steps:

S.No	Action	Inputs	Expected Output	Actual Output	Test Browser	Test Result	Test Comments
1	Launch application	https://www.facebook.com/	Facebook home	Facebook home	IE-11	Pass	[Priya 10/17/2017 11:44 AM]: Launch successful
2	Enter correct Email & Password and hit login button	Email id : test@xyz.com Password: ******	Login success	Login success	IE-11	Pass	[Priya 10/17/2017 11:45 AM]: Login successful

Principles of Software Testing

Principles of Software Testing

1. Testing shows the Presence of Defects

- The primary purpose is to find any defects that might cause the product failure.

- The testing team cannot say that the product is 100% error-free even though testing is done

- It reduces the number of undiscovered defects in the application.

- Therefore, design the needed test cases to find the defects as much as possible.

2. Exhaustive Testing is Impossible

- Sometimes it seems to be very hard to test all the modules and their features.

- So, instead of performing exhaustive testing they use some important testing criteria effects such as risk analysis and priorities modules.

- Then Product can be delivered on schedule.

Principles of Software Testing

3. Early Testing

- The testing team will have a deep understanding of the product as they were involved from the beginning of the requirement gathering and analysis phase.

- If errors are found in an initial stage of the development life cycle, then it will be easier to fix, save time and cost is also less.

- Otherwise, if they found late, then you need to change the whole system process.

4. Defect Clustering

- A small number of modules contain most of the defects detected.

- As per Pareto's Principal or 80-20 Rule, 80% of Defects are Caused by 20% of code.

- Testers focus on a small area of team; they may fail to miss the bugs from other areas.

Principles of Software Testing

5. Pesticide Paradox

- If same set of test cases used again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software.

- So, new and different tests are necessary to be written for the implementation of multiple parts of the software, which helps us to find more bugs.

6. Testing is context-dependent

- There are multiple fields such as e-commerce, commercial websites, banking applications etc. These projects and products include different elements, features & requirements.

- So, different types of products can be tested differently based on nature.

- Same test cases cannot be applied for different projects.

Principles of Software Testing

7. Absence of Errors Fallacy

- Once the application is completely tested and there are no bugs identified before the release, so we can say that the application is 99 percent bug-free.

- Finding and fixing defects doesn't help if the system is unusable or does not meet users' needs and expectations.

- **Absence of error fallacy** means identifying and fixing the bugs would not help if the application is impractical and not able to accomplish the client's requirements and needs.
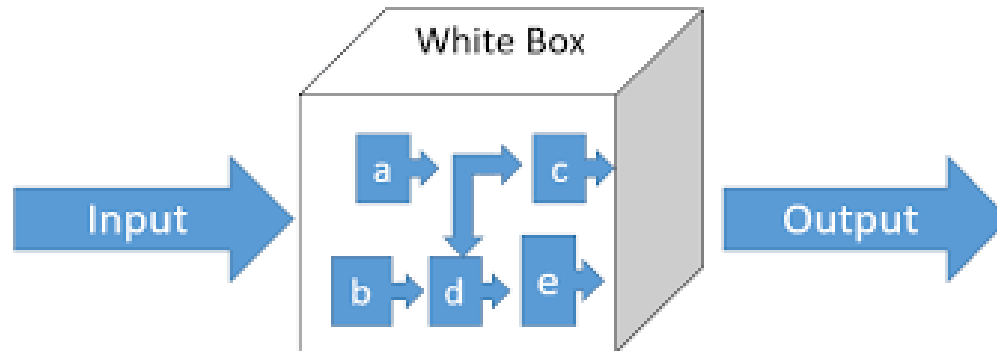
White Box Testing

- White box testing techniques analyze Internal design, Code structure & working_of the complete software product.

- It is also called Clear box testing. Open box testing. Transparent box testing, Code-based testing and Glass box testing.

- Tester has knowledge about the internal structure or the programming of the software.

- It is mostly done by software developers.

- Perform with Unit Testing & Integration Testing.

- **White Box Testing Tools:** EclEmma, Nunit, PyUnit, HTMLUnit, CppUnit
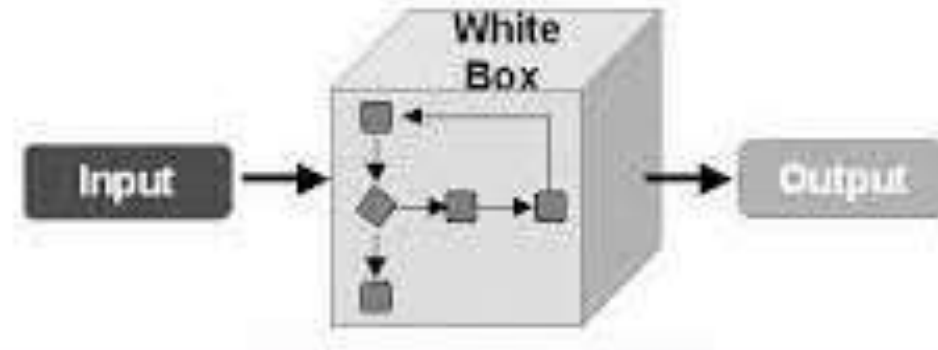
# What do you verify in White Box Testing?

✓Internal security of code

✓Poorly structured paths in the coding processes

✓The flow of specific inputs through the code

✓Loops, Decision Conditions

✓Each statement of your program

✓Functions & Modules on an individual basis

✓ Expected output

White Box Testing Techniques

1. Path Coverage / Testing:

- Testing is dependent on the program's control or flow structure.

- All possible paths are defined with entry to exit points in the system.
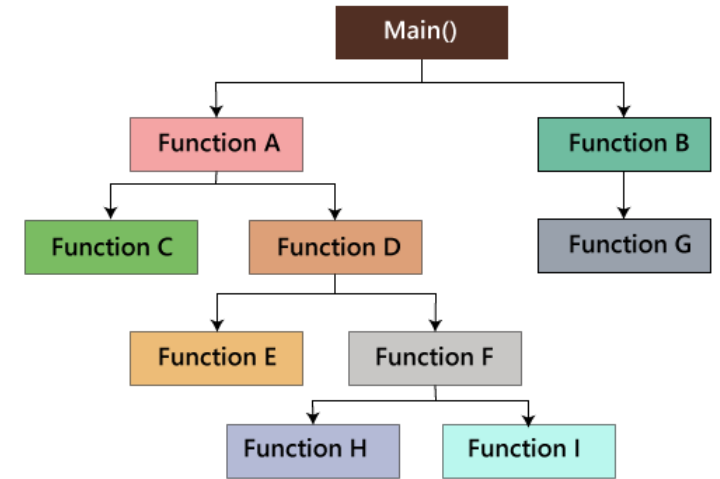
2. Loop Testing:

- There are loops such as While, For and Do-while loops etc.

- It check all Simple, Nested & Concatenated loops.

- It check ending condition if working correctly and if the size of the conditions is enough.
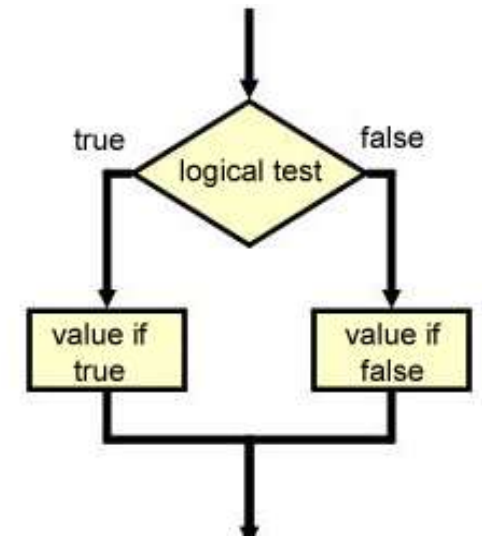
```
while(condition 1) {
    statement(s);
}
while(condition 2) {
    statement(s);
}
```
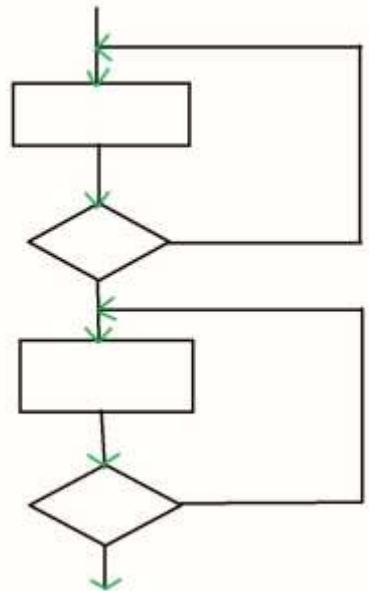
White Box Testing Techniques

3. Branch Coverage / Condition Testing:

- The logical conditions for every value are checked, whether it is true or false.
- This means that both the if and else conditions are verified.

4. Statement Coverage:

- Statement coverage is a testing technique in which tester traverse all statements in code.
- It ensure that each statement of the code is executed at least once.
- Hence each line of the code is verified

```
1▾ Prints (int a, int b) {
2    int result = a+ b;
3    If (result> 0)
4        Print ("Positive", result)
5    Else
6        Print ("Negative", result)
7    }
```

Advantages & Disadvantages of White Box Testing

➤Advantages of White Box Testing:

1. White box testing optimizes code so hidden errors can be identified.

2. Test cases of white box testing can be easily automated.

3. Early detection of errors that improve code quality.

4. It cover most path & code of the software.

 ➤Disadvantages of White Box Testing:

1. White box testing can be quite Complex, Expensive & Time-consuming.

2. Requires professional resources with detailed understanding of programming.

3. Missing functionalities cannot be detected.

4. Redesign of code needs test cases to be written again.

Black Box Testing

- In Black Box Testing, the functionalities of software applications are tested without having knowledge of Internal code structure. Implementation details and Internal paths.

- Tester gives input value to examine its functionality & checks whether function is giving expected output or not.

- If the function produces correct output, then it is passed in testing, otherwise failed.

- It is also known as Behavioral Testing. Functional & Closed box Testing.

- Perform by the Software Testers.

- **Black Box Testing Tools:** QTP, Selenium, Loadrunner & Jmeter

Types of Black Box Testing
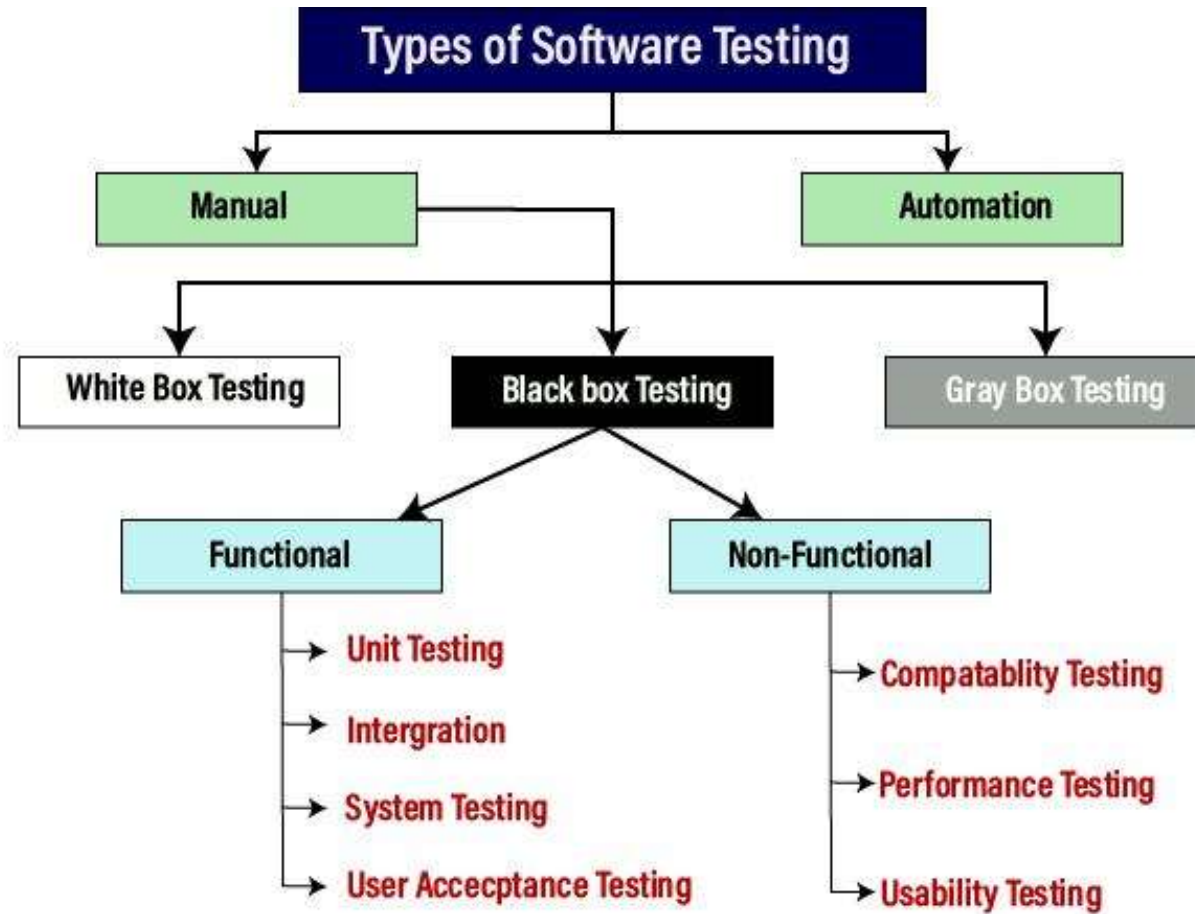
Black Box Testing Techniques

1. Equivalence Partitioning:

- Here, input values that provide to system are divided into different classes or groups based on its similarity in the outcome.

- Instead of using each & every input value, use any one value from the group to test outcome.

AGE `Enter Age` *Accepts value 18 to 56

EQUIVALENCE PARTITIONING		
Invalid	Valid	Invalid
<=17	18-56	>=57

2. Boundary Value Analysis:

- It test, boundary values are those that contain the upper and lower limit of a variable.

- It tests, while entering boundary value whether the software is producing correct output or not.

AGE `Enter Age` *Accepts value 18 to 56

BOUNDARY VALUE ANALYSIS		
Invalid (min -1)	Valid (min, +min, -max, max)	Invalid (max +1)
17	18, 19, 55, 56	57

Black Box Testing Techniques

3. Decision Table Testing:

• Various input combinations & their respective system behavior are captured in tabular form.

• Check logical relationship between two and more than two inputs. Ex. Gmail Account

| Email (condition1)	T	T	F	F
Password (condition2)	T	F	T	F
Expected Result (Action)	Account Page	Incorrect password	Incorrect email	Incorrect email

4. Error Guessing:

• It is based on the experience of the tester, where tester uses experience to guess the problematic areas of the software.

• **Examples**: Divide by zero, Handling null values in text fields, Accepting the Submit button without any value, File upload without attachment, File upload with less than or more than the limit size.

Black Box Testing Techniques

5. State Transition Testing:

- It is used to capture behavior of the software application when different input values are given to the same function.

- Applies to those types of applications that provide specific number of attempts to access application.
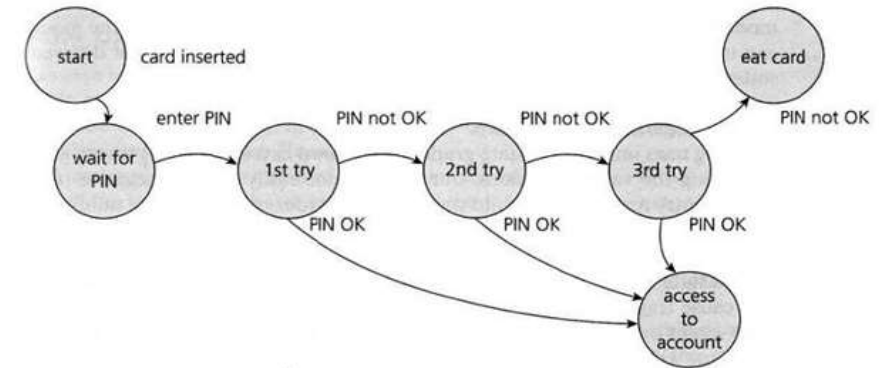
6. All Pairs Testing Techniques:

- It is used to test all the possible discrete combinations of values.

- This combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box etc.

Advantages & Disadvantages of Black Box Testing

➤**Advantages of Black Box Testing:**

1. The tester does not need to have more functional knowledge or programming skills.

2. It is efficient for implementing the tests in the larger system.

3. Tests are executed from the user's or client's point of view.

4. It is used in finding the ambiguity and contradictions in the functional specifications.

➤**Disadvantages of Black Box Testing:**

1. Without clear programming knowledge, test cases are difficult to implement.

2. It does not reveal the errors in the control structure.

3. Working with a large sample space of inputs can be exhaustive and consumes a lot of time.

Black Box VS White Box Testing

No.	Black Box Testing	White Box Testing
1	The internal structure & working of the software did not known to the tester. Focus on Input & Output of Product.	The internal structure & working of the software is known to the tester.
2	Is also known as Functional Testing, Data-driven testing, Closed-box testing & Behavioral Testing.	It is also known as Structural testing, Glass box testing, Code-based testing, and Transparent testing.
3	Less programming knowledge is required.	Requirement of complete programming knowledge.

Black Box VS White Box Testing

No.	Black Box Testing	White Box Testing
4	It is done at <u>higher levels of testing</u> that are system and acceptance testing.	It is done at <u>lower levels of testing</u> that are unit testing and integration testing.
5	Mainly performed by <u>Software Testers</u>.	Mainly performed by <u>Developers</u>.
6	<u>Less</u> time-consuming	<u>More</u> time-consuming
7	The base of this testing is <u>external expectations.</u>	The base of this testing is coding which is responsible for <u>internal working</u>.

Black Box VS White Box Testing

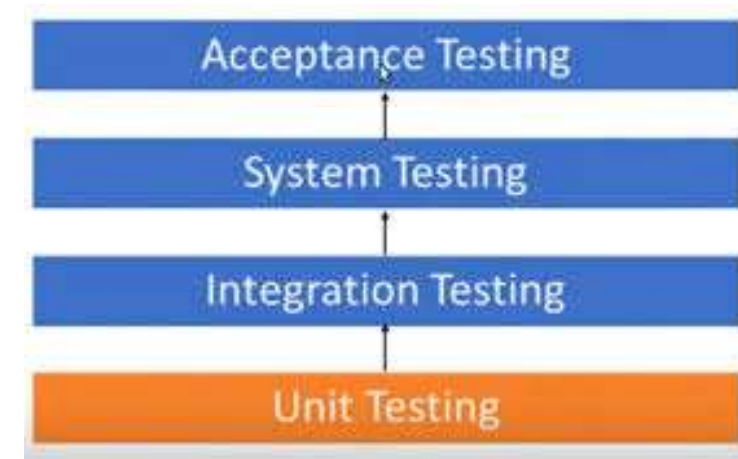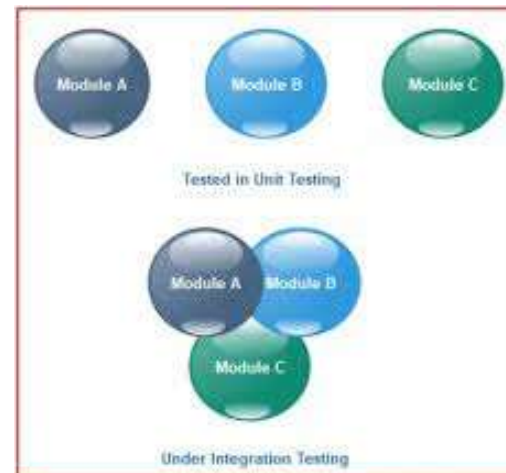| No.	Black Box Testing	White Box Testing
8	Its main objective is to fulfill customer need.	Its main objective is to check the code quality.
9	Defects are identified once the code is ready.	Defects are identified at early stage
10	Types of black-box testing: Functional & Non-Functional Testing and Regression testing.	Types of white box testing: Path testing, Loop testing and Condition testing.

Black Box VS White Box Testing

No.	Black Box Testing	White Box Testing
11	Can be done by <u>trial and error</u> ways and <u>methods.</u>	Data domains along with <u>inner or internal boundaries</u> can be better tested.
12	**Example:** Search something on Google by using keywords	**Example:** By input to check and verify loops

Unit Testing

- Unit testing is the first level of testing done before other remaining levels of the testing.

- In Unit Testing individual Units or Components of a software are tested.

- A Unit may be an individual function, method, procedure, module or object in program.

- Every module of project check one by one which is also called as components testing. It used white box testing techniques.

- It is done during coding phase of SDLC by the Software Developers.

- **Unit Testing Tools:** JUnit, NUnit, PHPUnit, EMMA & Jmockit

Tested in Unit Testing

Module A Module B Module C

Under Integration Testing

Acceptance Testing

System Testing

Integration Testing

Unit Testing

Why Unit Testing?

1. To understand correctness code & enables them to make changes quickly.

2. To test every function and procedures in project.

3. Help to fix bugs early in the development cycle and save costs.

4. It helps in the documentation.

5. It help with code re-use. Migrate both your code & tests to your new project.

6. Makes Faster & Efficient Software.

►**Unit Testing in the OO Context:**

• In program, There is different Packages & Classes which contains number of different operations like functions, methods, subclasses, derived classes, private, public & protected attributes etc.

Example of Unit Testing
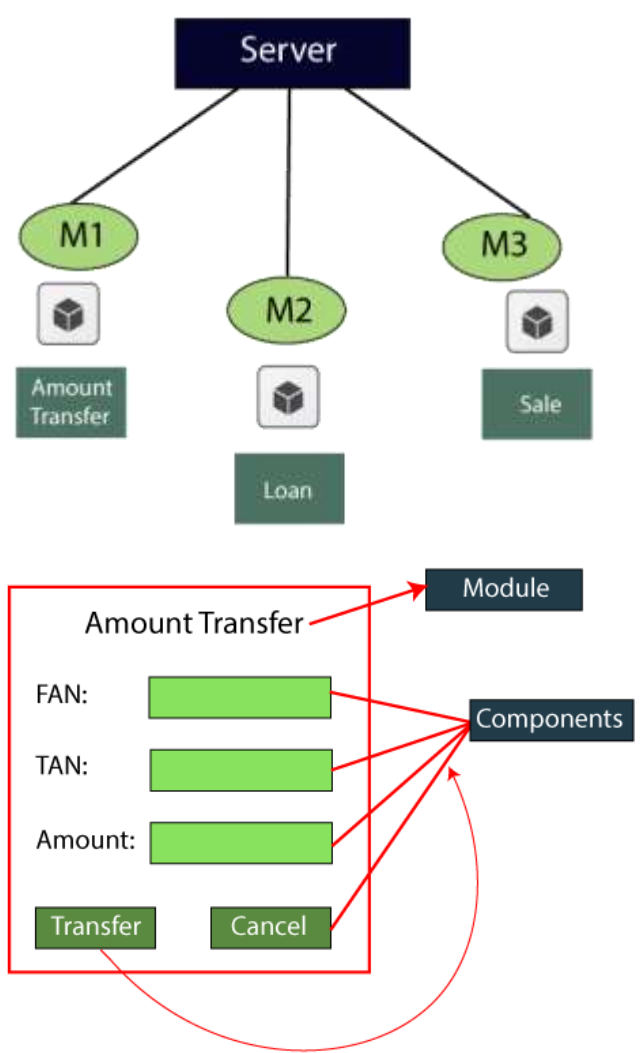
For From Account Number (FAN) & To Account Number (TAN)

Values	Description
1234	Account Number Accepted
4300	Error: Account Number Invalid
Blank	Error: Enter Account Number
Alphanumeric	Error: Enter digits only
Block Acc. No	Error Message

For Transfer Button

Description
Enter valid FAN value
Enter valid TAN value
Enter correct amount
Click on Transfer Button

For Cancel Button

Description
Enter value of FAN, TAN & Amount
Click on Cancel Button: All data should be clear

Advantages of Unit Testing

1.Due to the modular nature of the unit testing, Tester test parts of the project without waiting for others to be completed.

2. Developing team focuses on the functionality of the unit and how functionality should work.

3. It allows developers to detect and fix issues early in the development process, before they become larger and more difficult to fix.

4. It ensure that each unit of code meets the requirements, improving the overall quality of the software.

5. Unit testing enables developers to work faster and more efficiently

Disadvantages of Unit Testing

1. Unit testing requires more time and effort to create and maintain the test cases, especially for complex systems.

2. Unit testing may not be sufficient for testing interactions between units, as it only focuses on individual units.

3. Unit testing requires ongoing maintenance and updates, as the code and test cases must be kept up-to-date with changes to the software.
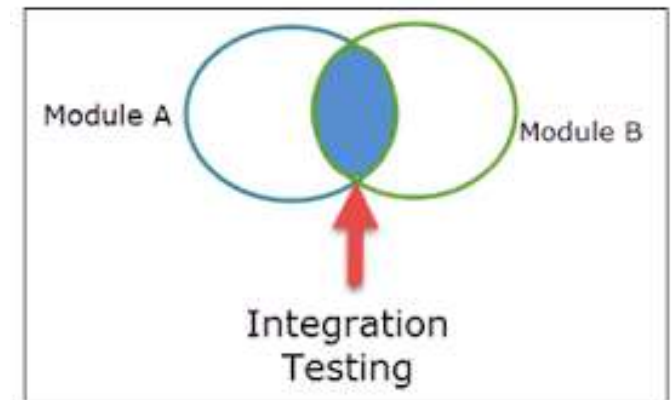
Integration Testing

- Integration testing is the second level of the software testing process comes after unit testing

- All modules may be coded by the different programmers or developers.

- Integrating various components or modules of an application & testing their behavior as a single unit or group called as Integration Testing.

- Also known as Thread testing or String testing.

- The goal of Integration Testing is to check correctness & communication among all modules.

➤**Integration Testing Tools:** Selenium, PyTest, Junit, Jasmine, Steam, Mockito etc.
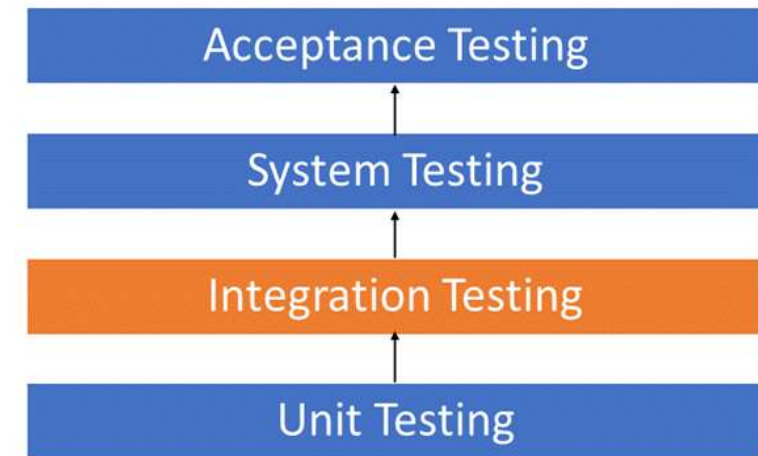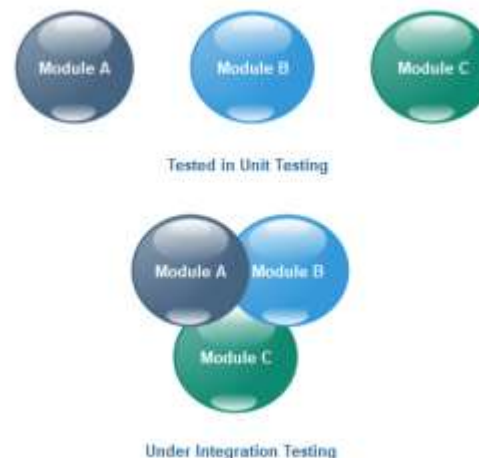
Why Integration Testing?

1. Each module is designed by different software developer whose programming logic may differ from developers of other modules, so integration testing becomes essential to determine the working of software modules.

2. To check the interaction of software modules with the database whether it is correct or not.

3. Customer requirements can be changed at the time of module development. These new requirements may not be tested at the unit testing hence integration testing becomes mandatory.

4. Incompatibility between modules of software could create errors.

5. To test hardware's compatibility with software.

Example of Integration Testing

After User 1 Login to Gmail Account

**User 1:** Click on Compose mail, Fill details & Send Mail to User 2.

If not send mail save in 'Draft' otherwise save in 'Sent Items'. Log out the Account.

After User 2 Login to Gmail Account User 2: Move to the 'Inbox' and verify that if the mail has reached. Open mail & check. If needed, Reply it. Log out the Account.

Compose Mail

Inbox
Compose mail
Sent Items
Trash
Spam
Contact
Folders
Logout

To

From

Subject

TEXT FIELD

☐ Save To Draft
☐ Add To Contact

SEND   CANCEL

Types of Integration Testing
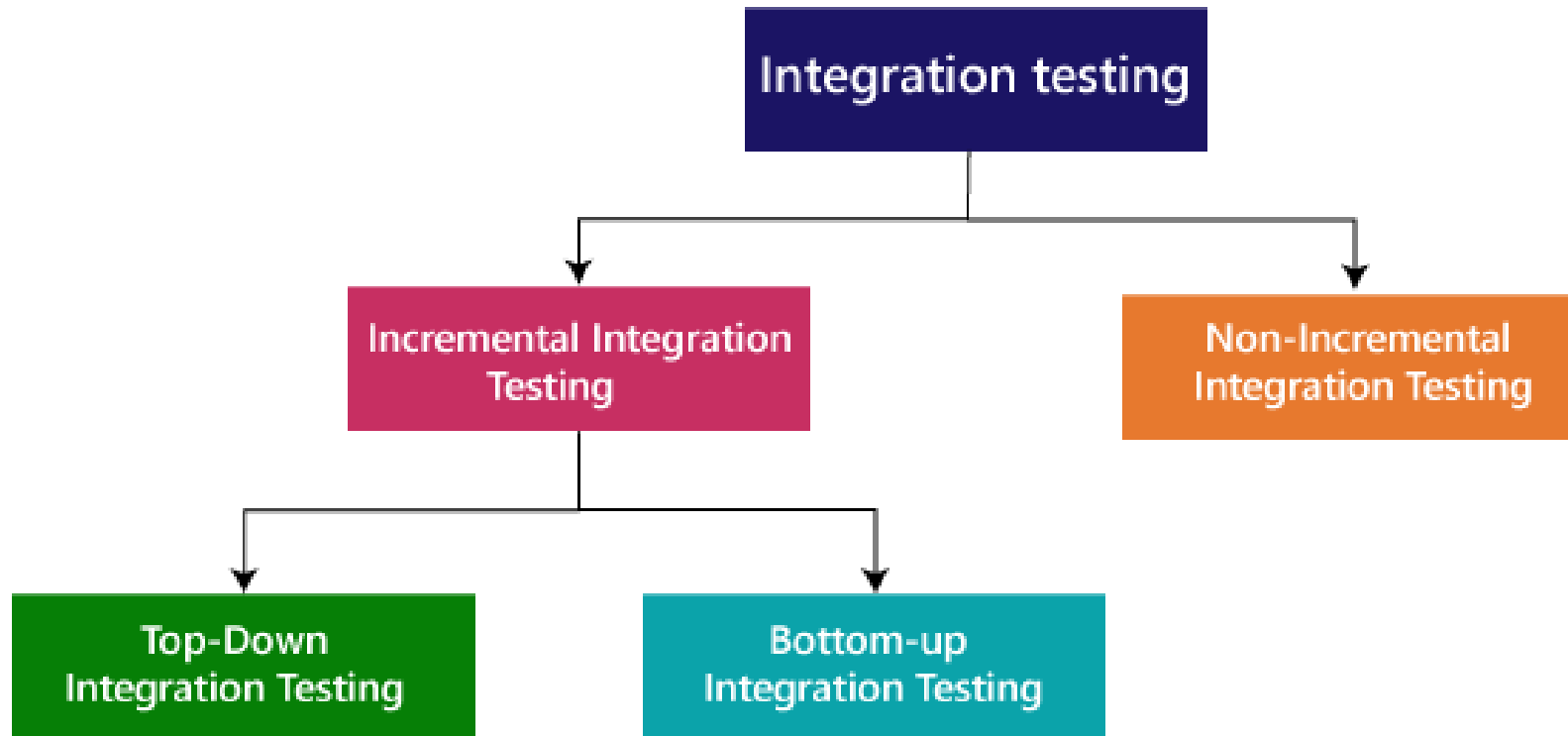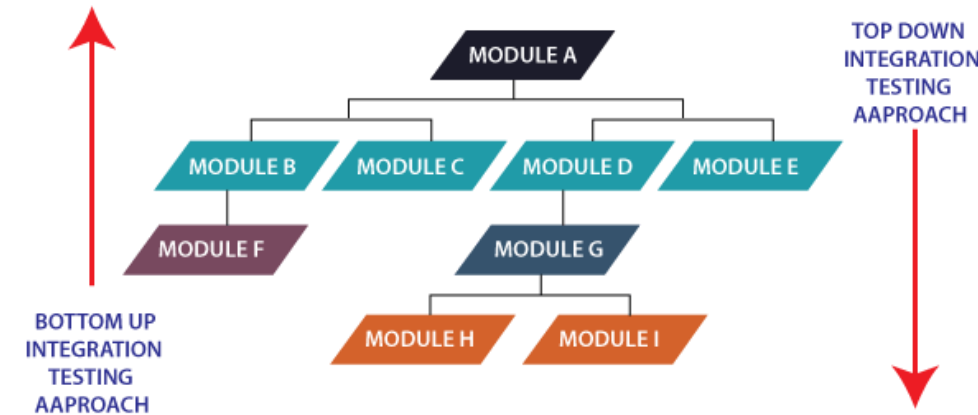
Types of Integration Testing

1. Incremental Integration Testing:

- Here, modules are e added in ascending order one by one or according to need.

- The selected modules must be logically related.

- They test data flow between modules & correctness of each functions.

- The process continues until the successful testing of all the modules.

- **It has two types: Top-Down & Bottom-Up**

Example:

Flipkart Application, Flow of testing application would like this:

Flipkart → Login → Home → Search → Add cart→ Payment → Logout

Types of Integration Testing

1.1 Top-Down Integration Testing:

- It starts by testing top-most modules & moving down to the lowest set of modules one-by-one & check control flow of system.

- As there is a possibility that the lower-level modules might not have been developed while top modules are tested.

- It will use Stubs as a dummy programs which act as a substitutes for the missing models in the testing.

- Here, Critical modules are tested on priority & Major design flaws are detected as early as possible.
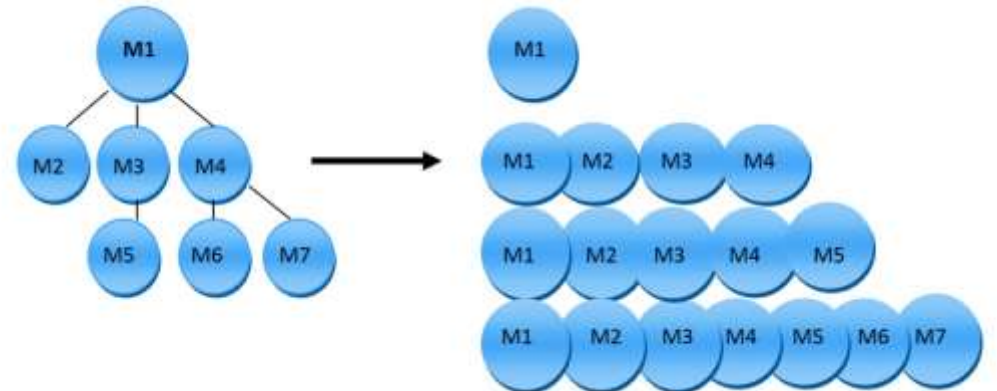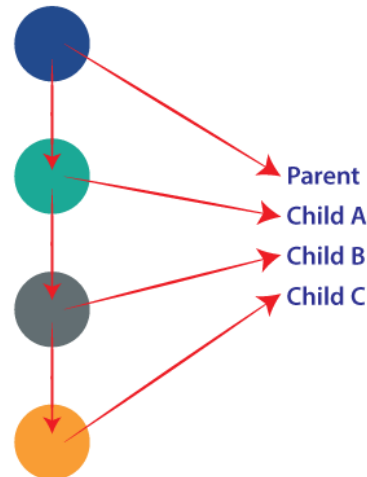
Types of Integration Testing

1.2 Bottom-Top Integration Testing:

- It starts with testing the lowest units of the application and moving up one-by-one.

- Here testing takes place from the bottom of the control flow to upwards.

- Again, it's possible that the higher-level modules might not have been developed by the time lower modules are tested.

- It will use Driver as a dummy programs which act as a substitutes for the missing models in the testing.

- Here, several disjoint subsystems can be tested simultaneously & easy to observe test result
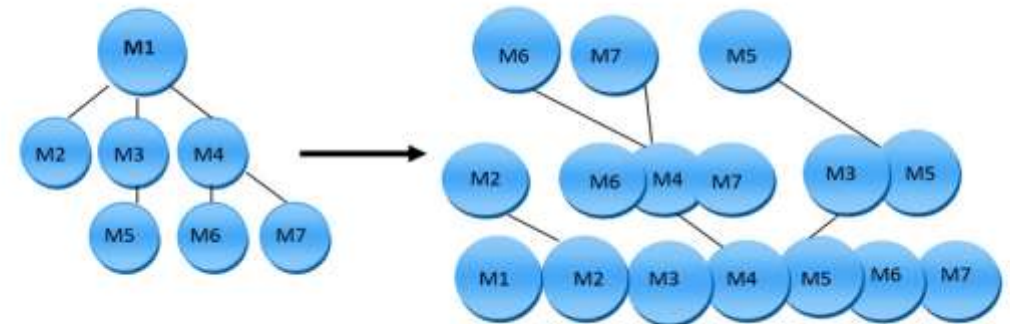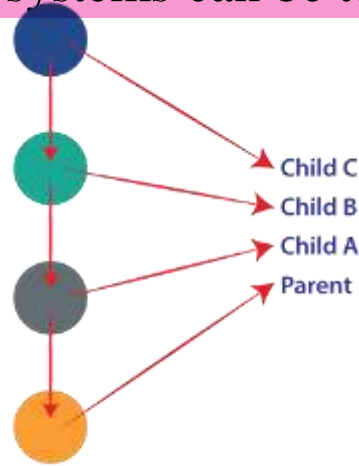
Types of Integration Testing

2. Non-Incremental Integration Testing/Big Bang Testing

- In this, once all the modules are developed and tested individually, they are integrated once and tested together at once. No parent & child concept.

- The only advantage of this type of testing is that it is very much suitable for smaller systems.

- If an error is found, then it is very difficult to solve because it belong to any of the modules being integrated.

- The goal is to verify the overall functionality of the system and to identify any integration problems that arise when all components are combined.
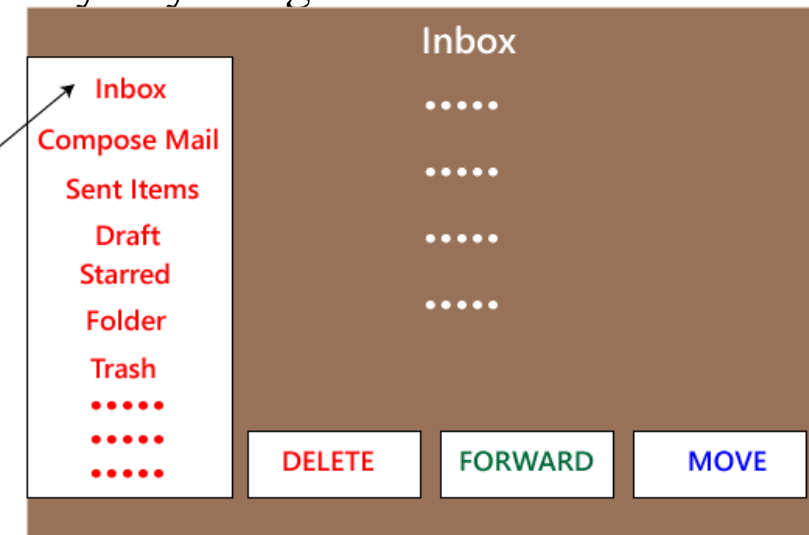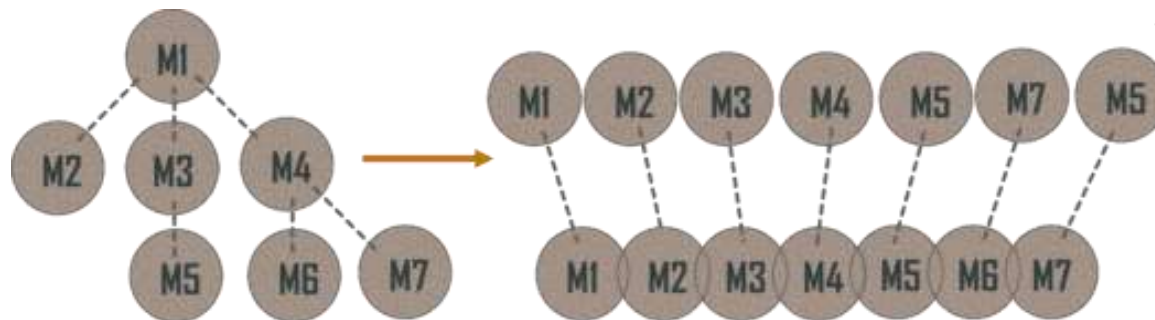
System Testing

- System Testing is validates the complete and fully integrated software product.
- The purpose of a system test is to evaluate the end-to-end system specifications.
- It's a type of black box testing perform by Quality Assurance Team in Testing phase.
- It focus on the Functionality, Accuracy. Quality. Expected Output & Overall behavior of an application rather than the inner working of system.
- **System Testing Tools:** Selenium, LoadRunner, Jmeter, Microsoft Test Manager, SoapUI
- The choice of tool depends on various factors like Technology used. Size of the project, Budget & Testing requirements etc.
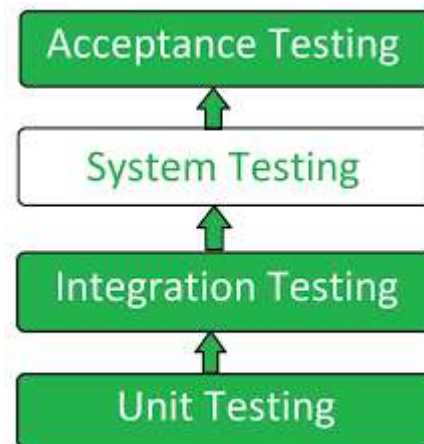
Importance of System Testing

1. Improved Product Quality: It test product can successfully work across different platforms and environments.

2. Error Reduction: It verifies a system's code & functionality as per requirements, so errors that aren't detected during integration and unit testing can be exposed during system testing.

3. Cost Savings: Conducting timely & continuous system testing reduces unexpected costs & project delays.

4. Security: They ensure that the tested system doesn't contain potential vulnerabilities or bugs that can put end users' system data at risk.

5. Customer Satisfaction: It builds customer confidence & improves overall user experience.

6. Software Performance: It track complete system performance. Also help to understand changes in a system's performance & behavior, such as memory consumption, central processing unit utilization etc. Inform developers to take proactive action.

Types of System Testing

1. Performance Testing

2. Load Testing

3. Usability Testing

4. Regression Testing

5. Migration Testing

6. Functional Testing

7. Recovery Testing

8. Stress Testing

9. Software & Hardware Testing

Types of System Testing

1. Performance Testing:

It measures Speed, Load time, Stability, Reliability & Response times of the system under various conditions.

2. Load Testing:

It determine how system or software performs under a real-life extreme load. Such as Throughput, Number of users etc.

3. Usability Testing:

It evaluate system is easy to use and functional for the end user. Such as User error rates, Task success rates, Time takes a user to complete a task & User satisfaction etc.

System Testing

Types of System Testing

4. Regression Testing:

- It ensure that any changes done during the development process have not introduced a new defects or bugs.

- Also ensure that old defects or bugs will not exist on the addition of new software over the time.

5. Migration Testing:

- It ensure that if system needs to be modified in new infrastructure so it should be modified without any issue.

6. Functional Testing:

- Tester find out any missing function in the system & make list of it.

- It can be added during functional testing and should improve quality of the system.

Types of System Testing

7. Recovery Testing:

- It ensure that system is capable of recovering from certain system errors, crashes and failures.

8. Stress Testing:

- It ensure that robustness of the system under the various loads & capacities.

9. Software & Hardware Testing:

- This testing of the system check hardware and software compatibility & interaction.

- The hardware configuration must be compatible with the software to run it without any issue.

System Testing Example

Test Case Type	Description	Test Step	Expected Result	Status
Functionality	Area should accommodate up to 20 characters	Input up to 20 characters	All 20 characters in the request should be appropriate	Pass or Fail
Security	Verify password rules are working	Create a new password in accordance with rules	The user's password will be accepted if it adheres to the rules	Pass or Fail
Usability	Ensure all links are working properly	Have users click on various links on the page	Links will take users to another web page according to the on-page URL	Pass or Fail

Acceptance Testing

- It is a <mark>final level of Software testing</mark> before making the system available for actual use.

- It is type of <mark>Black box testing</mark> & <mark>performed by the End user</mark> or Client.

- Here, end-user or customer uses the application to find out if the software meets the user's expectations and works as expected.

- It performed on user/ <mark>live environment</mark> or in real time scenarios.

- Also called as User Acceptance Testing (UAT), Functional Acceptance Testing (FAT), End User Testing or Red Box Testing (RBT).

- **Acceptance Testing Tools:** <mark>Fitness Tools, Watir</mark> etc.
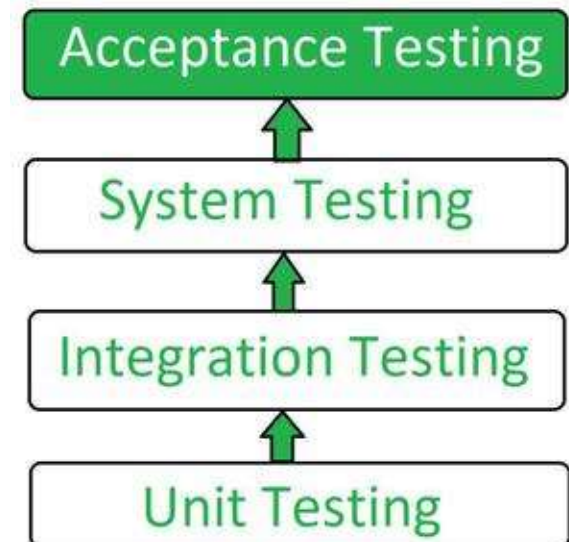
Importance of Acceptance Testing

1. Helps identify bugs that may have been left during the development process.

2. It helps to identify that the end product works according to client's or user's expectations or requirements.

3. It brings confidence and satisfaction to the clients as they are directly involved in the testing process.

4. Helps to deliver the end product without any bugs.

5. To satisfy all the functionalities that mentioned in SRS document.

Types of Acceptance Testing

1. User Acceptance Testing (UAT):

- It performed from the end-user point of view.

- Client check whether the software product is working as per the requirements of the user correctly before moving it into the production environment.

- This testing is performed in a different production-like set-up environment.

- It does not focus on errors & bugs mainly focus on functionality required by the clients.

- This testing is also known as End-User- Testing.

Types of Acceptance Testing

2. Business Acceptance Testing (BAT):

- It check whether software product is able to meet the business requirements & operational need or not as per real world.

- It mainly focuses on the business risk & financial factors, which is one of the challenging things in the changing market conditions and the advancement of technologies.

3. Regulations Acceptance Testing (RAT):

- It check whether software product developed with the rules & regulations of the country or region where it is getting released.

- If any product is released even in the case of violation of rules and regulations, the software product owner will be considered responsible.

Alpha Testing VS Beta Testing

No	Alpha Testing	Beta Testing
1	It performed by a team of highly skilled testers who are usually the internal employee of the organization.	It performed by clients or end-users in client location or in real-time environment.
2	It involves both white box & black-box techniques.	It uses only black-box testing.
3	Reliability or Security not performed in-depth, Focus on bugs, errors etc.	Reliability, Security & Robustness checked during beta testing.
4	Long execution cycles maybe require.	Only a few weeks are required for the execution.

Alpha Testing VS Beta Testing

No	Alpha Testing	Beta Testing
5	Critical issues can be identified by developers immediately.	Most of the issues or feedback is collecting from user will be implemented for the future versions of the product.
6	It focus on the product's quality before going to beta testing.	It focus on product's quality & ensures that the product is ready for real-time users.
7	It performed nearly the end of the software development.	It is a final test before shipping a product to the customers.

Example of Verification & Validation

➤ **What's App Application:**

Verification Process	Validation Process
It means **"Are we implementing the software right?"**	It means **"Are we implemented the right software?"**
Developer developed: ✓ Chatting Functionality then Verify it. ✓ Status Functionality then Verify it. ✓ Audio Call Functionality then Verify it. ✓ Video Call Functionality then Verify it. ✓ Share Photos Functionality then Verify it.	Tester Validate complete developed product at same time. ✓ Validate Chatting, Status, Audio Call, Video Call, Share Photos functionality at a same time.

Verification VS Validation

No	Verification	Validation
1	It means "Are we implementing the software right?"	It means "Are we implemented the right software?"
2	It comes before validation	It comes after verification
3	It is known as Static Testing.	It is known as Dynamic Testing.
4	It is executed by Quality assurance team or Developers.	It is executed by the Testing team.
5	It includes checking documents, design, code and program.	It includes testing and validating the actual product
6	Whether the software conforms to specification is checked	It checks whether the software meets the requirements and expectations of a customer
7	It does not involve executing the code	It always involves executing the code

Verification VS Validation

No	Verification	Validation
8	It finds bugs early in the development cycle	It can find bugs that the verification process can not catch
9	It includes different methods like Inspections, Reviews & Walkthroughs.	It includes testing like Functional, System, Integration & User acceptance testing.
10	Developer can verify that the inputs follow the outputs or not.	Tester can validate that the user accepts the product or not.

What is Defect or Bug?

- A **Defect or Bug** is an errors in an application that is created during building or designing software and due to which software starts to show abnormal behaviors during its use.

- The variation between the actual results and expected results is known as Defect.

➤**Defect/Bug Life Cycle:**

- In software testing is the specific set of states that defect or bug goes through in its entire life.

- The purpose of Defect life cycle is to easily coordinate and communicate current status of defect make the defect fixing process systematic and efficient.

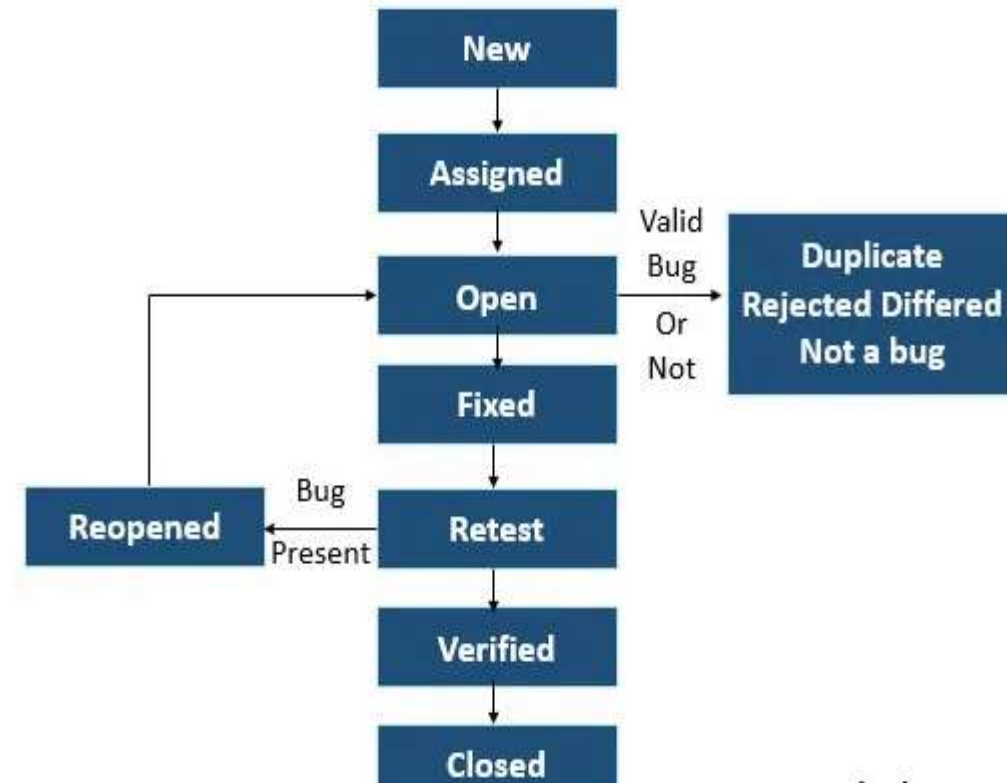➤Performed by: Developer & Tester

➤Tools: JIRA, Trac, Redmine etc.

Defect Life Cycle

1. NEW:

- Tester identified Defect.
- Tester send proper Defect document to Development team.

2. ASSIGNED:

- Defect is assigned to Developer team.

Defect Life Cycle

3. OPEN:

- Developer team works on Defect for fixing the issues.

- If developer team feels that defect is not appropriate then it is transferred to either 'Duplicate', 'Rejected', or 'Deferred' state.

- **Duplicate:** Defect is repeated or send twice.

- **Rejected:** Defect is invalid.

- **Deferred:** Defect is not of a high priority and is expected to get fixed in the next release.
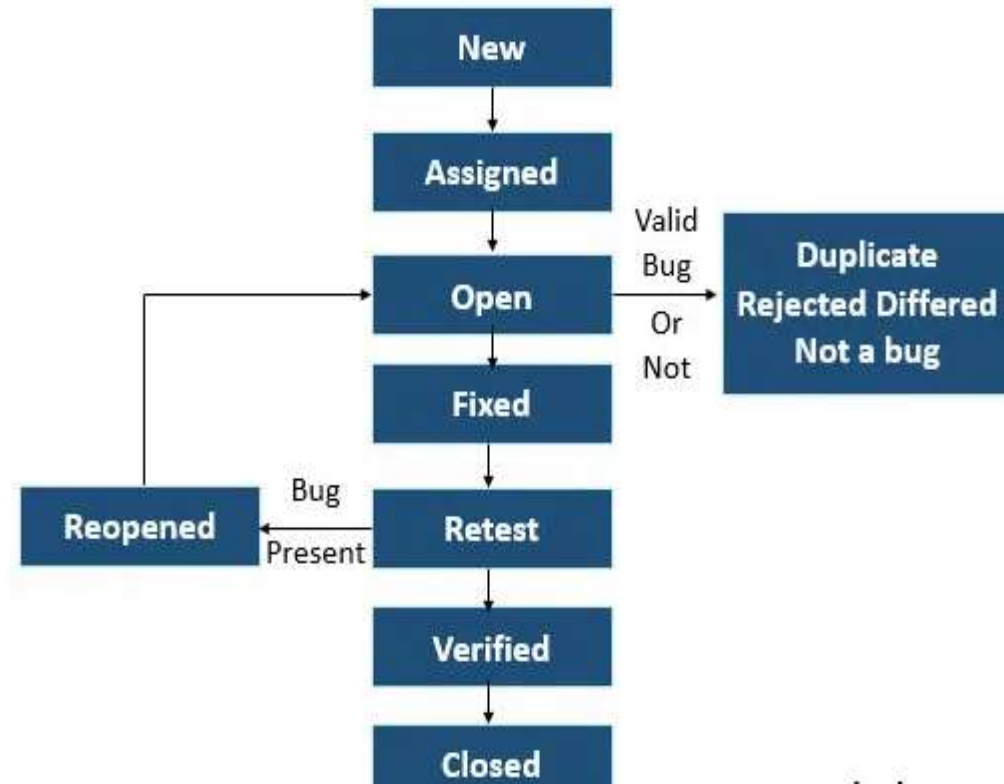
Defect Life Cycle

- **4. FIXED:**

- Developer takes necessary actions on coding to fix the Defect.

- The defect will remove from the application.

5. RETEST:

- Testing team to check whether the bug has fixed by the developer or not.
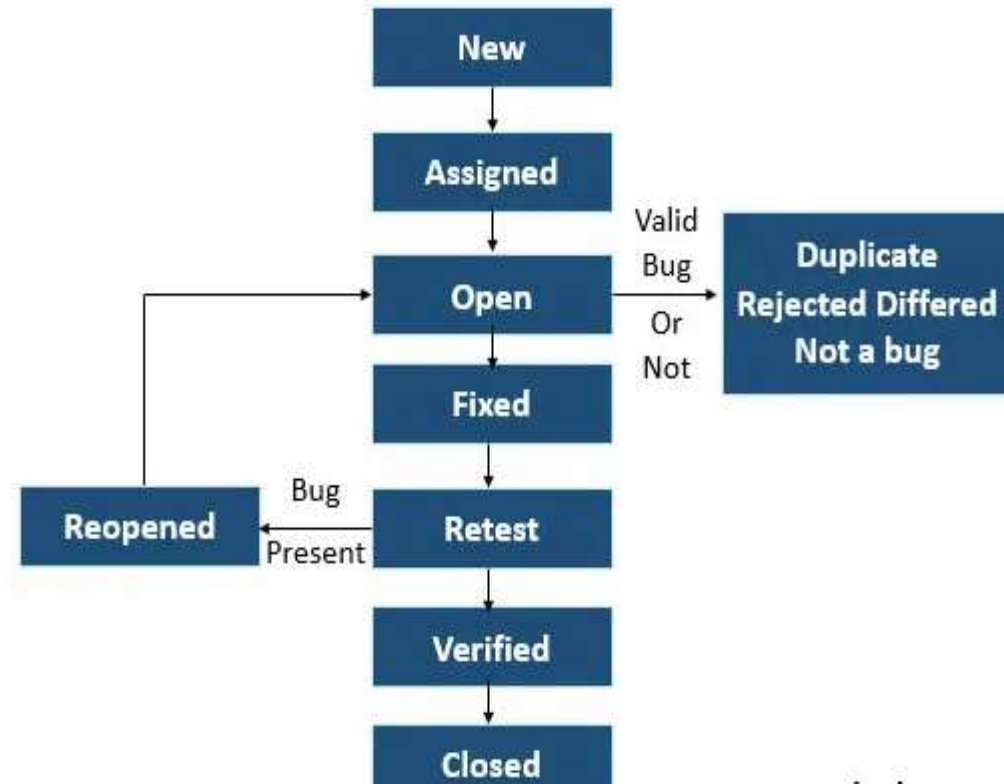
Defect Life Cycle

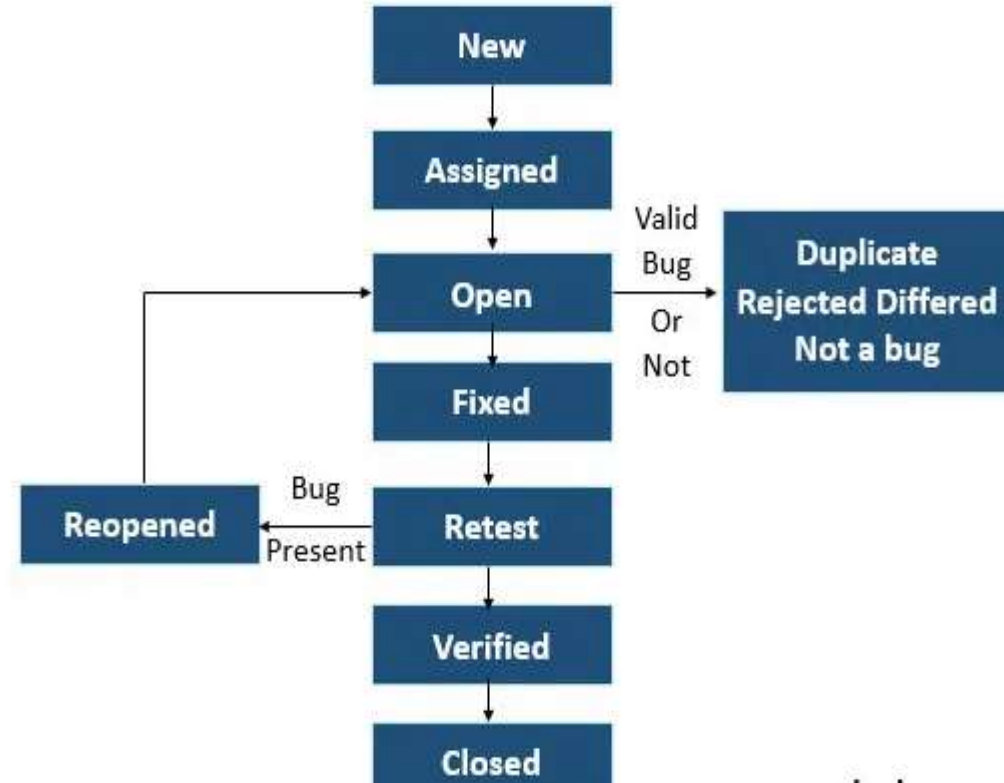## 6. REOPENED:

- If Defect still exists even after Developer team has fixed the bug.

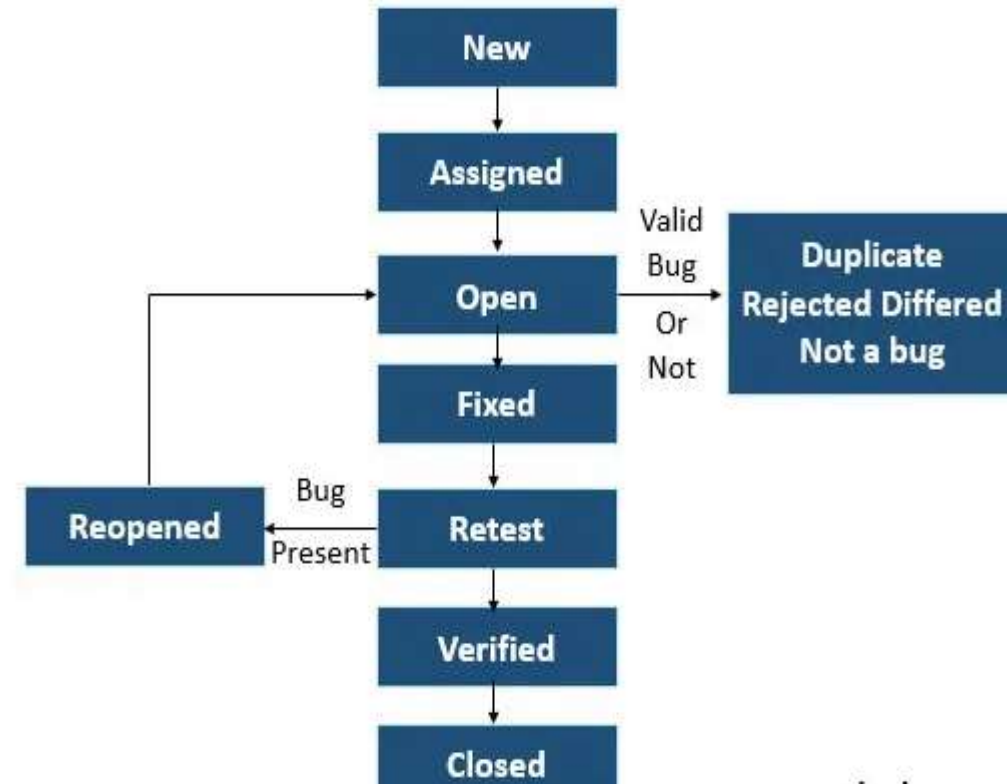- The lifecycle restarts.

7. VERIFIED:

- The tester re-tests Defect after it got fixed by Developer team.

- If tester does not find any kind of Defect, then status assigned is verified.

Defect Life Cycle

8. CLOSED:

- Once the Defect has been verified as fixed.
- The Testing team closes the issue.

Testing VS Debugging

No	Testing	Debugging
1	Testing is the process to find bugs and errors.	Debugging is the process of correcting the bugs found during testing.
2	It is done by the Tester.	It is done by either Programmer or Developer.
3	Testing can be manual or automated.	Debugging is always manual.
4	It is based on different testing levels i.e. unit testing, integration testing, system testing, etc.	Debugging is based on different types of bugs.
5	Programming knowledge is not required to perform the testing process.	Detail programming language require.

Testing VS Debugging

No	Testing	Debugging
6	Software testing is the vital phase of SDLC.	It is not a part of SDLC because it occurs as a subset of Testing.
7	Testing is initiated after the coding phase or code is written.	Debugging commences with the execution of a test case.
8	Tester can plan, design & implement the testing process.	The debugging process cannot be planed.
9	Testing is composed of the validation and verification of software.	The developers will logically evaluates and removes the software errors.

Software Quality

- The quality of software can be defined as the ability of the software to function as per user requirement.

- When it comes to software products it must satisfy all the functionalities written down in the SRS document.

- **Software Quality include:**

1. Good Design: Good visualization design to attract users.

2. Durability: The software work without any issue for a long period of time.

3. Consistency: Software perform consistently different platform and other devices.

4. Maintainability: Capture and fix bugs quickly. New features are added easily.

5. Value for money: Customer & companies who make this app should feel that the money spent on this app has not to waste.

Software Quality Dimensions / Parameters

1. Maintainability: The ease with which software can be modified (adding features, enhancing features, fixing bugs, etc.)

2. Portability: The ability of software to be transferred easily from one location to another.

3. Functionality: The ability of software to carry out the functions as specified or desired.

4. Performance: The speed at which software performs under a particular load.

5. Compatibility: The suitability of software for use in different environments like different devices, operating systems and browsers.

6. Usability: The degree of software's ease of use.

7. Reliability: The ability of software to perform a required function under stated conditions without any errors.

8. Security: The extent of protection of software against unauthorized access, invasion of privacy, theft, loss of data etc. Ex. OTP

Factors affect on Software Quality

1. Product Operation Factors:

Correctness, Reliability, Efficiency, Integrity, Usability.

2. Product Revision Factors:

Maintainability, Flexibility, Testability.

3. Product Transition Factors:

Portability, Reusability, Interoperability.

Software Quality Metrics

- SQM ensures that the software product is of highest quality and standard.

1. Customer Problem Metrics:

- Measuring the problems encountered by the customers while using the product.
- PUM Total problems reported by a customer + Total number of license months

2. Customer Satisfaction Metrics:

It deals with overall quality of product & how much a customer is satisfied with that product.

It is measured by Very Satisfied, Satisfied. Neutral, Dissatisfied, Very Dissatisfied.

3. Software Maintenance Metrics:

- After completion of Development & Testing product release in market.
- During this interval. How many defect arrived at customer environment?

Software Quality Management

- Software Quality Management (SQM) refers to the complete process that ensures a software product is develop as per national and international standards like ANSI, IEEE and ISO.

➤**Need of Software Quality Management:**

1. Delivering high-quality products on time.

2. Increases stakeholder faith on product & company.

3. High-quality products always ensure customer satisfaction.

➤**Activities of Software Quality Management:**

How to Achieve Software Quality?

Quality Assurance:

- It assure that system meets specified requirements and customer expectations.

- It defines standards and methodologies for successful development process.

- It assure Correctness, Efficiency, Flexibility, Maintainability, Portability, Usability etc.

Quality Control:

- It focuses on to achieve & fulfill quality parameters or quality goals as per customer requirements.

- It focus on deliver product on time with accurate cost.

Quality Planning:

- Select applicable procedures and standards for a particular project and modify as required to develop a quality plan

Quality Assurance VS Quality Control

No	Quality Assurance	Quality Control
1	It is a procedure that focuses on providing assurance that quality requested will be achieved.	Quality Control is a procedure that focuses on fulfilling the quality requested.
2	In order to meet the customer requirements, QA defines standards and methodologies	QC confirms that the standards are followed while working on the product
3	QA is process oriented.	QC is product oriented.
4	QA aims to prevent the defect	QC aims to identify and fix defects
5	It's a Proactive measure	It's a Reactive measure
6	It is performed before Quality Control	It is performed only after QA activity is done
7	It does not involve executing the program	It always involves executing a program

Quality Assurance VS Quality Control

No	Quality Assurance	Quality Control
8	QA involves in <u>full software development life cycle</u>	QC involves in <u>full software testing life cycle</u>
9	<u>All team members</u> are responsible for QA.	<u>Testing team</u> is responsible for QC.
10	It is the procedure to <u>create the deliverables</u>	It is the procedure to <u>verify that deliverables</u>
11	<u>Less time</u>-consuming activity	<u>More time</u>-consuming activity
12	QA ensures that everything is executed in the right way, and that is why it falls under <u>verification activity</u>	QC ensures that whatever we have done is as per the requirement, and that is why it falls under <u>validation activity</u>