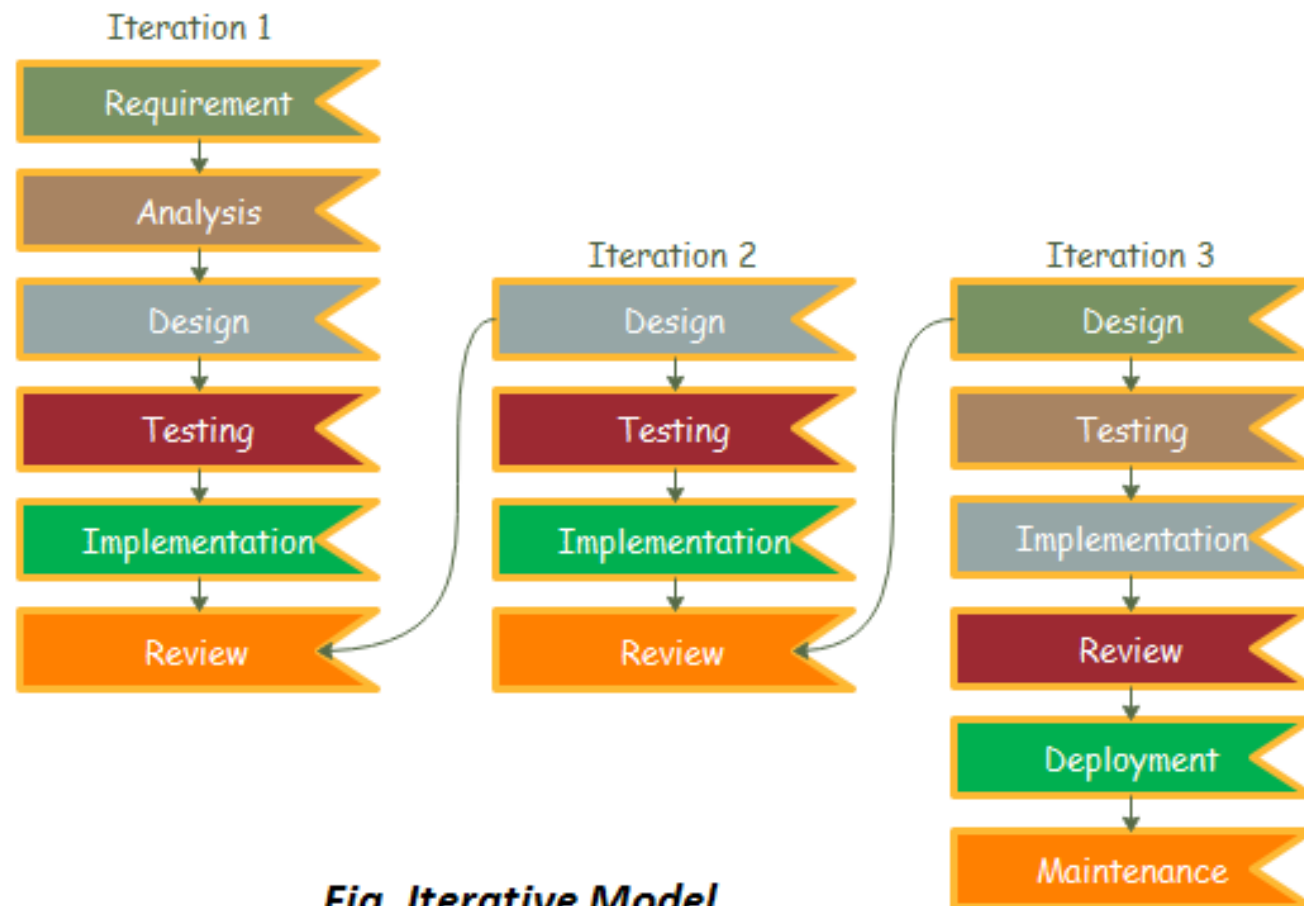


Iterative model

In this Model, you can start with some of the software specifications and develop the first version of the software. After the first version if there is a need to change the software, then a new version of the software is created with a new iteration. Every release of the Iterative Model finishes in an exact and fixed period that is called iteration.

The Iterative Model allows the accessing earlier phases, in which the variations made respectively. The final output of the project renewed at the end of the Software Development Life Cycle (SDLC) process.



When to use the Iterative Model?

- When requirements are defined clearly and easy to understand.
- When the software application is large.
- When there is a requirement of changes in future.

Advantage of Iterative Model

- Testing and debugging during smaller iteration is easy.
- A Parallel development can plan.
- It is easily acceptable to ever-changing needs of the project.
- Risks are identified and resolved during iteration.
- Limited time spent on documentation and extra time on designing.

Disadvantage of Iterative Model

- It is not suitable for smaller projects.
- More Resources may be required.
- Design can be changed again and again because of imperfect requirements.
- Requirement changes can cause over budget.
- Project completion date not confirmed because of changing requirements.

Incremental Model

- Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle.
- In this model, each module goes through the requirements, design, implementation and testing phases

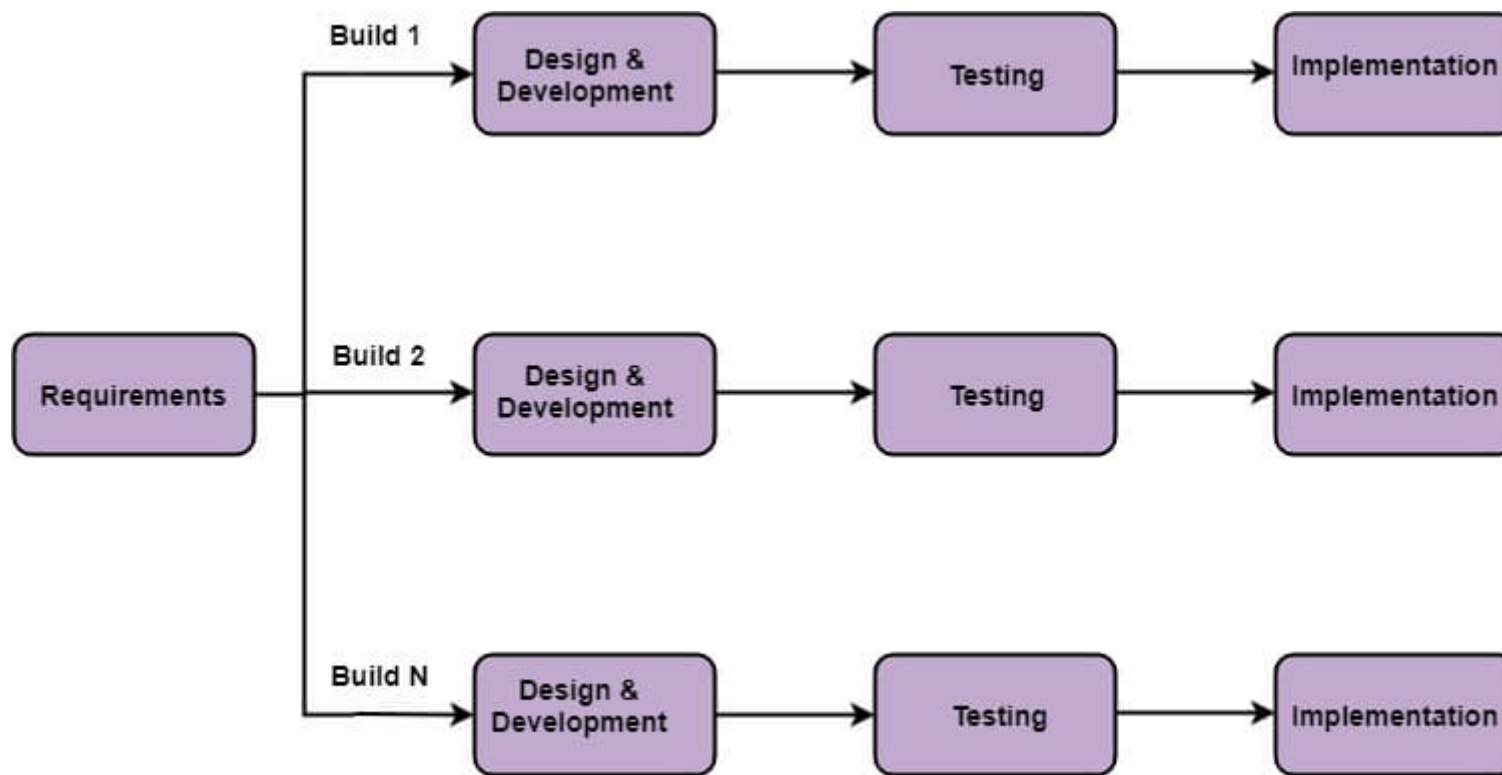


Fig: Incremental Model

When we use the Incremental Model?

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

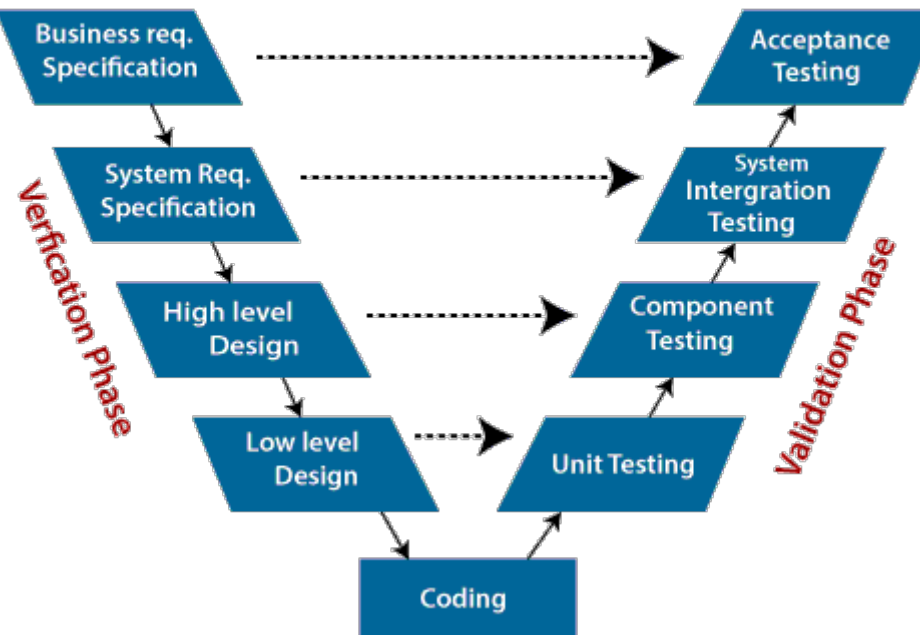
V- Model

- V-Model also referred to as the Verification and Validation Model(V-V).
- In this, each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model.
- Testing is planned in parallel with a corresponding stage of development.

V- Model

Developer's life Cycle

Tester's Life Cycle



Verification and validation

- Verification:
 - It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements meet.
Are we building the product right?
- Validation:
 - It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.
Are we building the right product?
- V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation process is joined by coding phase in V-shape. Thus it is known as V-Model

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when technical resources are available with essential technical expertise.

Advantage of V-Model

- Easy to Understand.
- Testing Methods like planning, test designing happens well before coding.
- This saves a lot of time. Hence a higher chance of success over the waterfall model.
- Works well for small plans where requirements are easily understood.

Disadvantage of V-Model

- Very rigid and least flexible.
- Not a good for a complex project.
- Software is developed during the implementation stage, so no early prototypes of the software are produced.
- If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

Agile

- Agile process model" refers to a software development approach based on iterative development
- Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.
- Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks
- Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

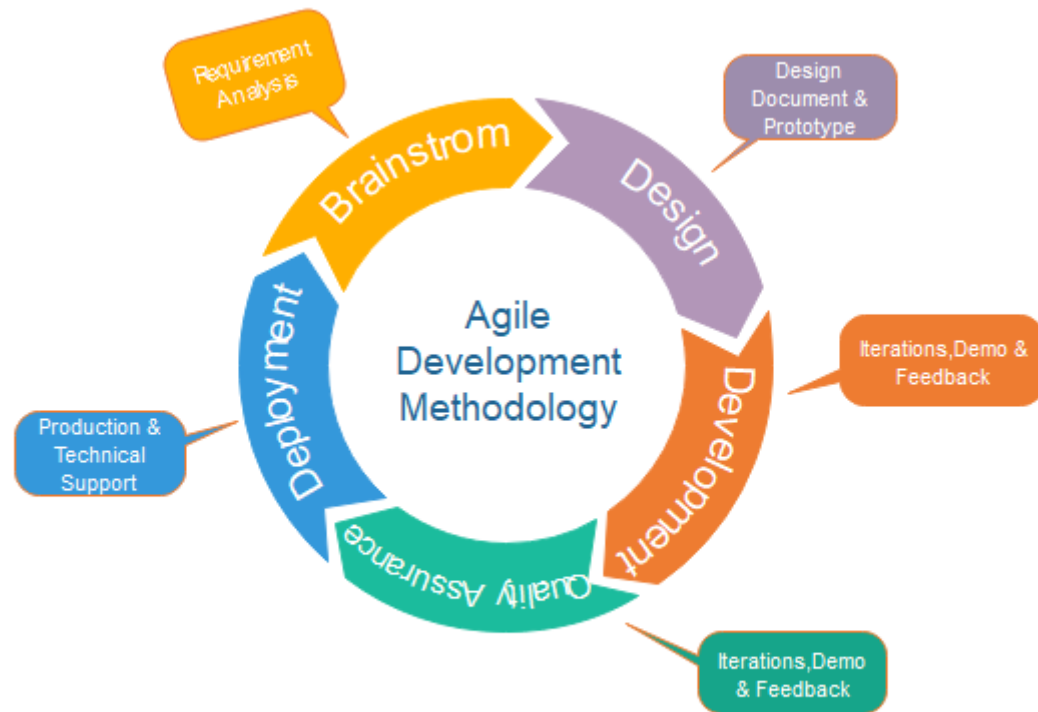


Fig. Agile Model

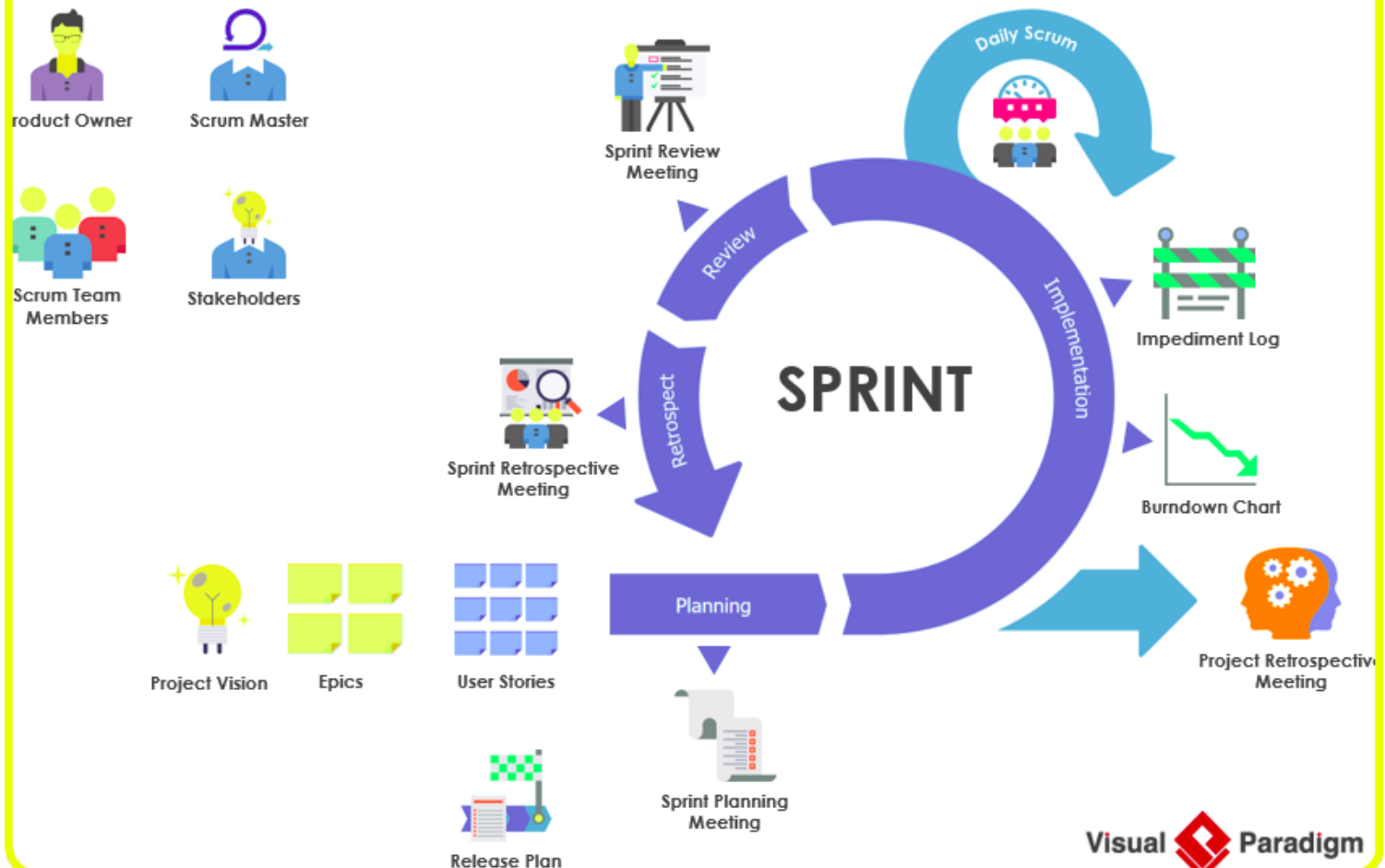
Agile Testing Methods

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

Scrum

- SCRUM is an agile development process focused primarily on ways to manage tasks in team-based development conditions.
- There are three roles in it, and their responsibilities are:
 - Scrum Master: The scrum can set up the master team, arrange the meeting and remove obstacles for the process
 - Product owner: The product owner makes the product backlog, prioritizes the delay and is responsible for the distribution of functionality on each repetition.
 - Scrum Team: The team manages its work and organizes the work to complete the sprint or cycle.

The Agile – Scrum Framework



Major Termonologies

Roles	Artifacts	Ceremonies
<ul style="list-style-type: none">• Product owner• Development team• Scrum master	<ul style="list-style-type: none">• Increment• Product backlog• Sprint backlog	<ul style="list-style-type: none">• Sprint planning• Sprint review• Sprint retrospective• Daily scrum

When to use the Agile Model?

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

Advantages of Agile Model

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

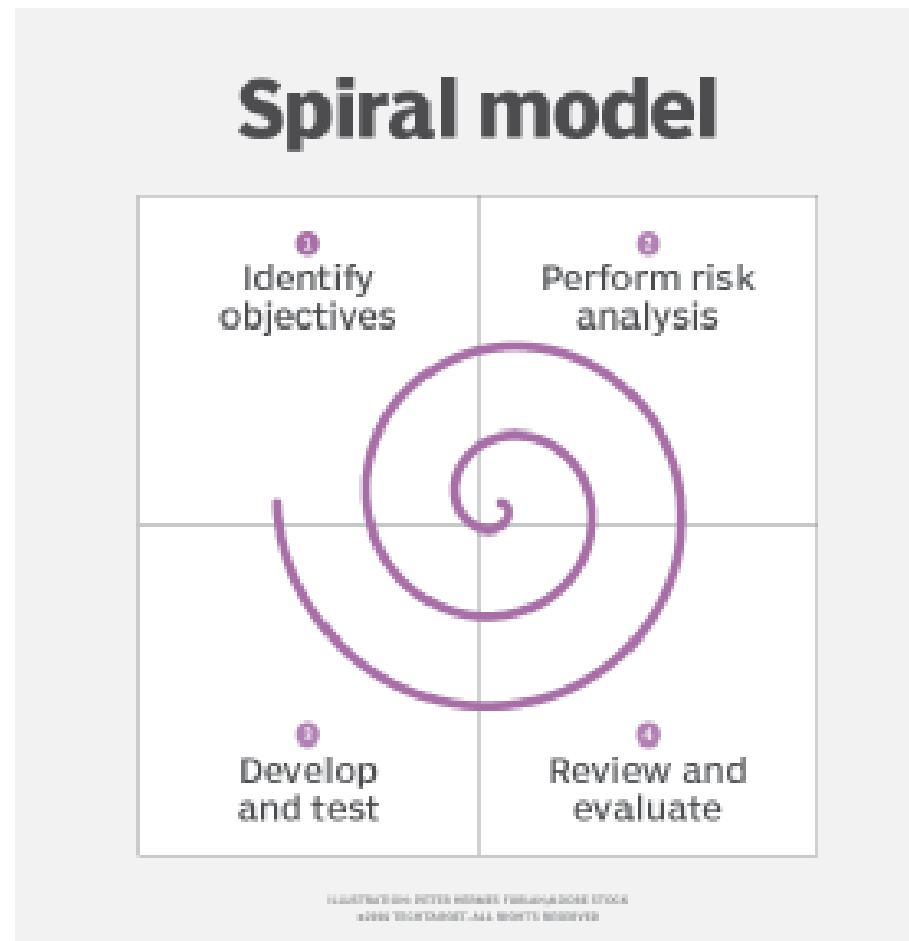
Disadvantages of Agile Model

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

Spiral model

- The spiral methodology merges the possibility of iterative improvement with some of the controlled parts of the waterfall model.
- This methodology is a fusion of the iterative development model and the waterfall model.

Spiral model design



Phases of spirall

- Planning phase- BRD and SRS
- Risk Analysis- Identify risk and create prototype
- Engineering phase -Software development and Testing at each phase
- Evaluation phase- Allow customer to evaluate the output of project to date before the project

Advantages

- Software is produced early in the software life cycle.
- Risk handling is one of important advantages of the Spiral model, it is best development model to follow due to the risk analysis and risk handling at every phase.
- Flexibility in requirements. In this model, we can easily change requirements at later phases and can be incorporated accurately. Also, additional Functionality can be added at a later date.
- It is good for large and complex projects.
- It is good for customer satisfaction. We can involve customers in the development of products at early phase of the software development. Also, software is produced early in the software life cycle.
- Strong approval and documentation control.
- It is suitable for high risk projects, where business needs may be unstable. A highly customized product can be developed using this.

Disadvantages

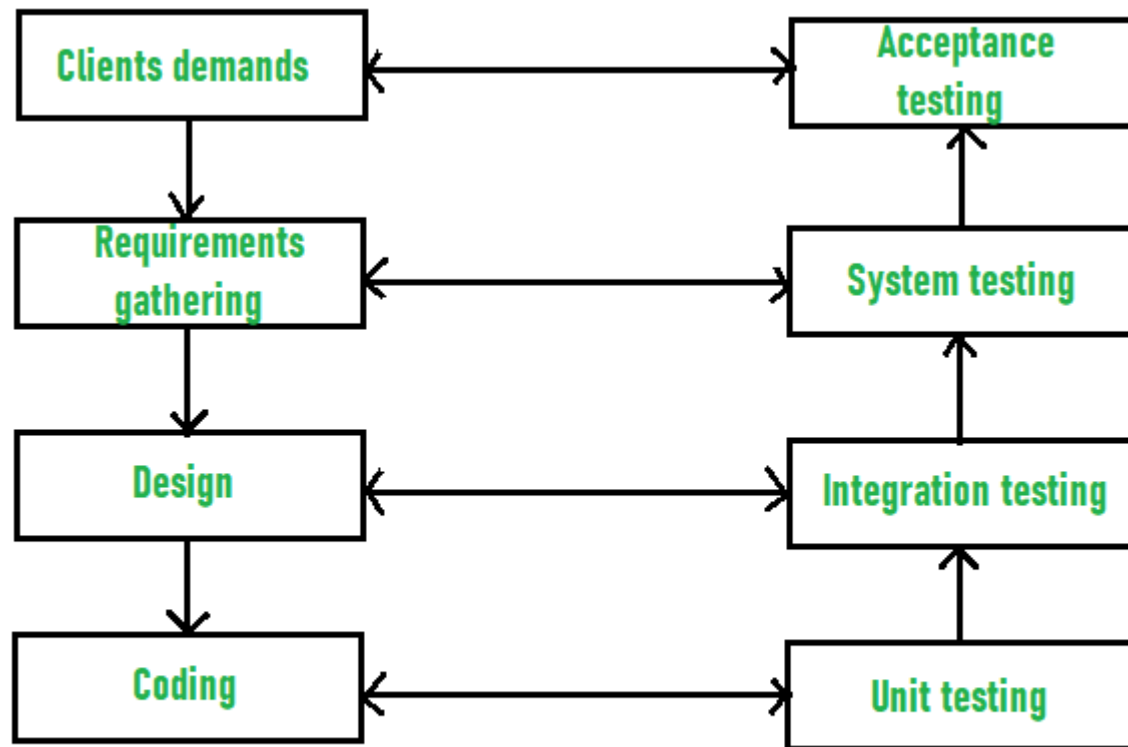
- It is not suitable for small projects as it is expensive.
- It is much more complex than other SDLC models. Process is complex.
- Too much dependable on Risk Analysis and requires highly specific expertise.
- Difficulty in time management. As the number of phases is unknown at the start of the project, so time estimation is very difficult.
- Spiral may go on indefinitely.
- End of the project may not be known early.
- It is not suitable for low risk projects.
- May be hard to define objective, verifiable milestones. Large numbers of intermediate stages require excessive documentation.

When to use

- When cost and risk Analysis is important
- Medium to high risk project
- Long term commitment
- Complex reqd significant change expected

Level Of Testing

- There are generally four recognized levels of testing
- Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test.



LEVELS OF TESTING

Unit Testing

- Unit testing ensures that each part of the code developed in a component delivers the desired output. In unit testing, developers only look at the specification for a component
- Code is thoroughly tested standalone before progressing to another unit.

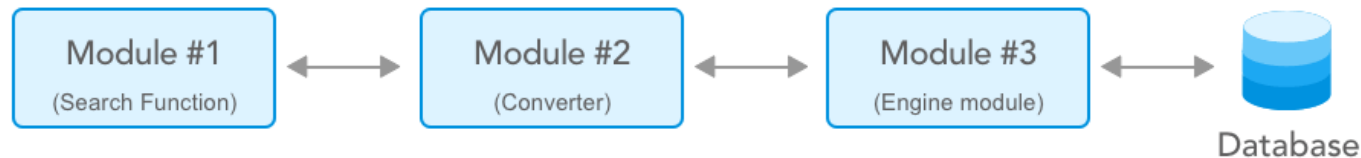


Integration Testing

- Integration testing is performed to test the modules which are working fine individually and do not show bugs when integrated. It is the most common functional testing type
- Generally, developers build different modules of the system/software simultaneously and don't focus on others. They perform extensive black and white box functional verification, commonly known as unit tests, on the individual modules. Integration tests cause data and operational commands to flow between modules which means that they have to act as parts of a whole system rather than individual components
- In integration testing, we check the interfacing between the inter-connected components.

Integration Testing

- **Example-** search functionality in the e-commerce site
 - Module #1: This is the search box visible to users where they can enter text and click the search button.
 - Module #2: It's a converter or in simple terms program which converts entered text into XML.
 - Module #3: This is called Engine module which sends XML data to the database.
 - Module #4: Database



System Testing

- System testing is testing conducted on a complete, integrated system to evaluate its compliance with the specified requirements.
- After the completion of the integration testing, the product is passed for system testing.
- System Testing is very important because it verifies that the application meets functional and non functional requirements that were set by the stakeholder
- In system testing, we check the system as a whole.

User Acceptance Testing

- User acceptance testing (UAT) is the last phase of the software testing process.
- In UAT actual software/app users test the software to make sure it can handle required tasks in real-world scenarios
- it is performed at the time of product delivery to stakeholders as a final checkpoint among all functional testing types.
- Users do not use 'Test to Break' approach while doing user acceptance testing. Rather, UAT is a measure of how good your application performs in normal scenarios.

Different type of testing

- Smoke testing
- Regression testing
- Sanity testing
- Re-testing
- Functional Testing
- Non-Functional Testing

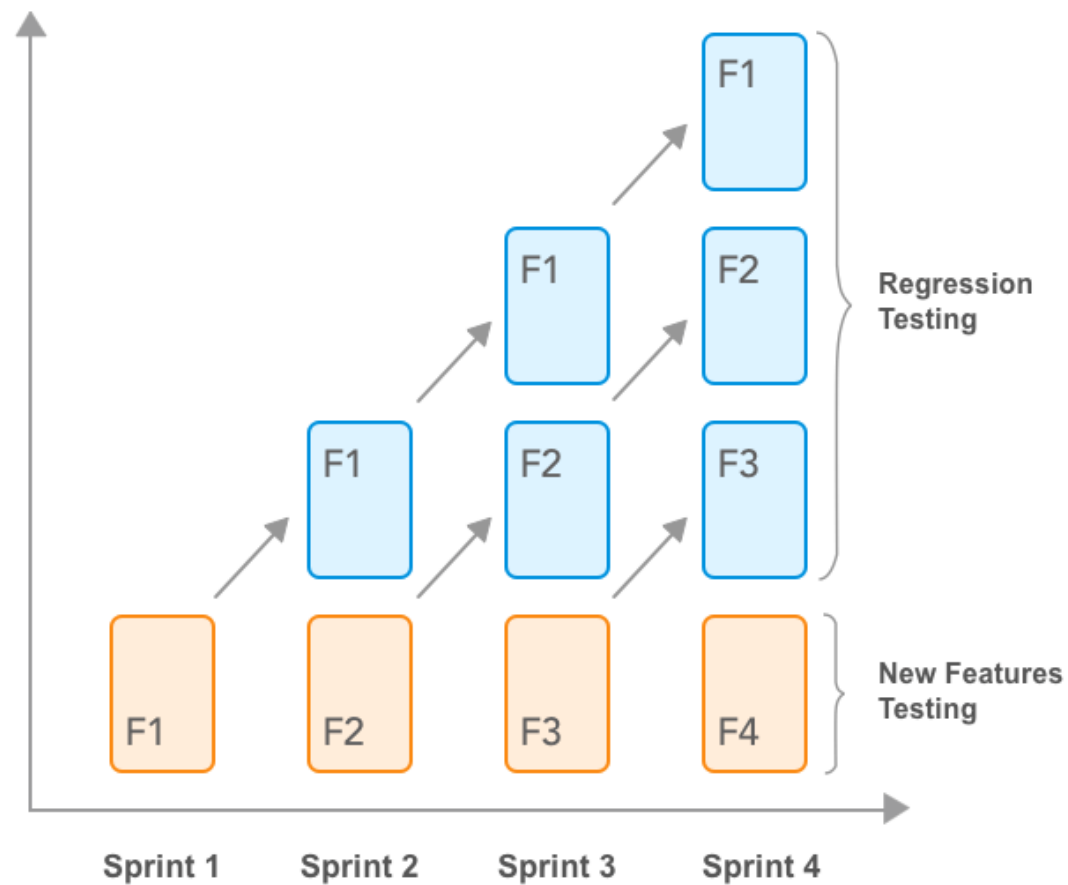
Smoke testing

- Smoke testing is performed on the 'new' build given by developers to QA team to verify if the basic functionalities are working or not.
- This should be the first test to be done on any new build.
- In smoke testing, the test cases chosen cover the most important functionality or component of the system.
- The objective is not to perform exhaustive testing, but to verify that the critical functionality of the system is working fine.

Regression testing

- Regression testing is performed to make sure that a change or addition hasn't broken any of the existing functionality
- Its purpose is to find bugs that may have been accidentally introduced into the existing build
- Regression testing can become a challenge for the testers as well. Here are some of the reasons:
 - The Number of test cases in the regression suite increases with each new feature.
 - Sometimes, the execution of the entire regression test suite becomes difficult due to time and budget constraints.
 - Minimizing the test suite while achieving maximum test coverage is not a cake walk.
 - Determination of frequency of Regression Tests after every modification or every build update or after a bunch of bug fixes is always a challenge.

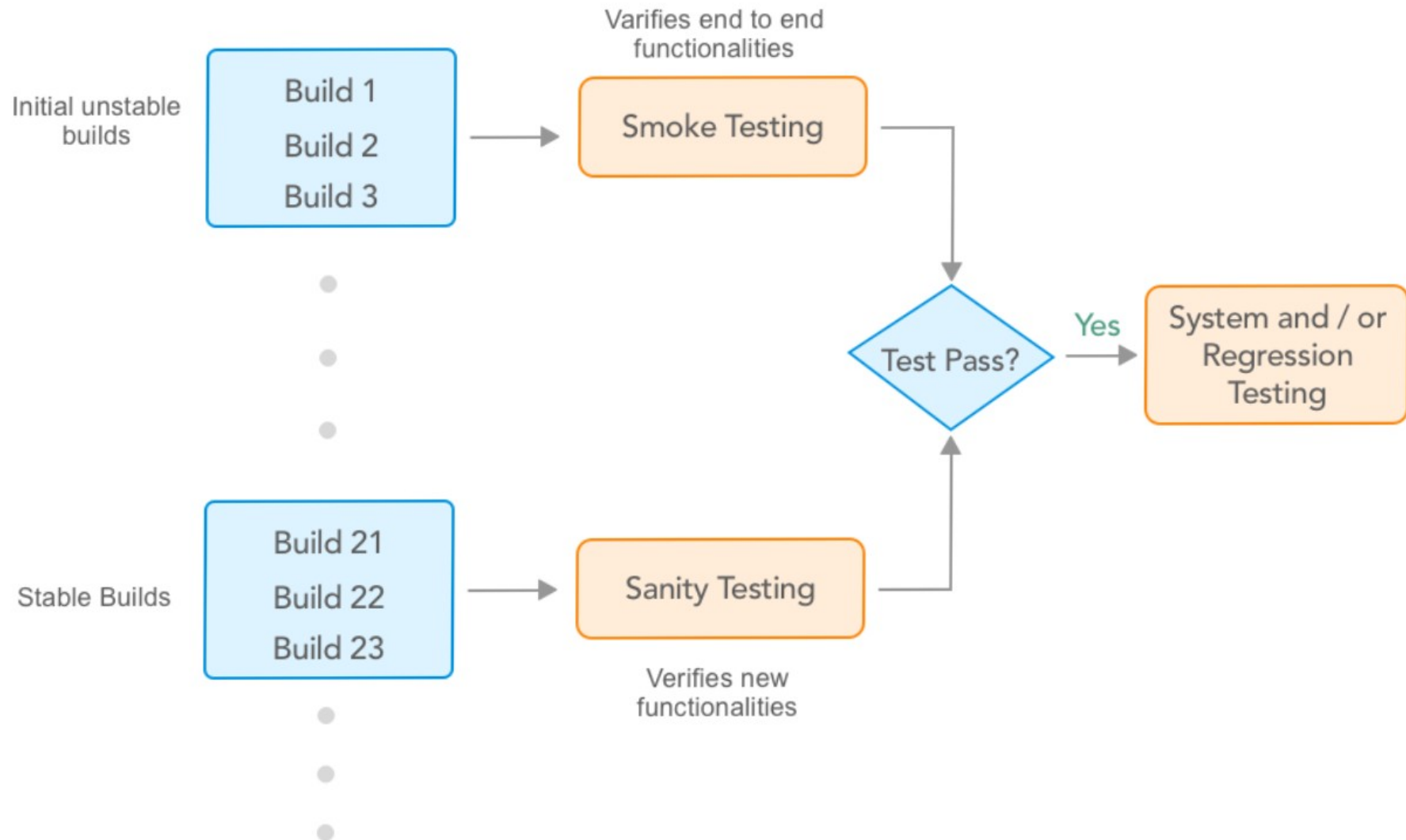
Regression Testing



Sanity Testing

- When a new build is received with minor modifications, instead of running a thorough regression test suite we perform a sanity test.
- It is a subset of Regression testing
- Group of test cases executed that are related to the changes made to the product

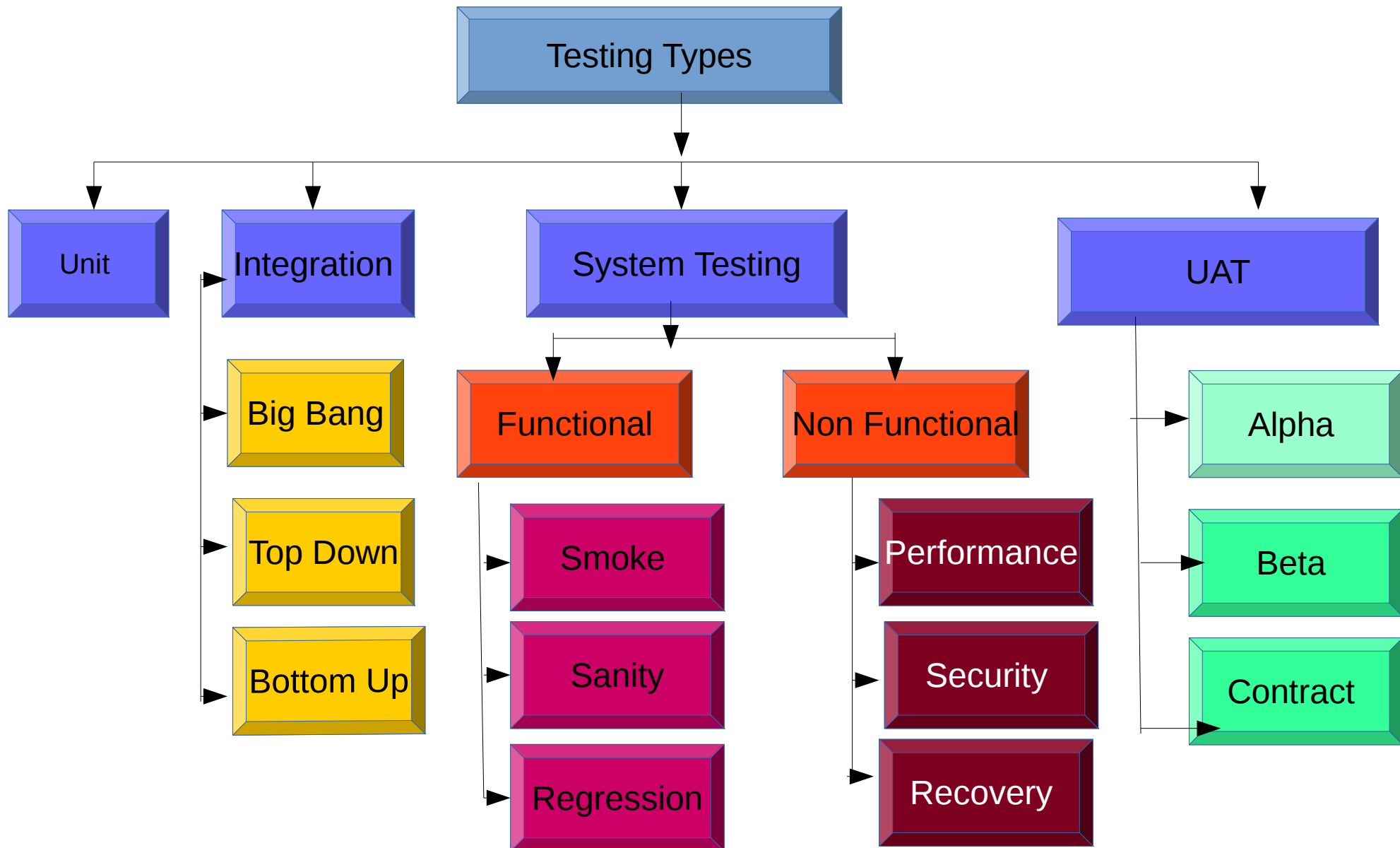
Confusion between Sanity and Smoke



Re-testing

- Re-testing is executing a previously failed test against new software to check if the defect is fixed
- After a defect has been fixed, re-testing is performed to check the scenario under the same environmental conditions.

Testing Type Overview

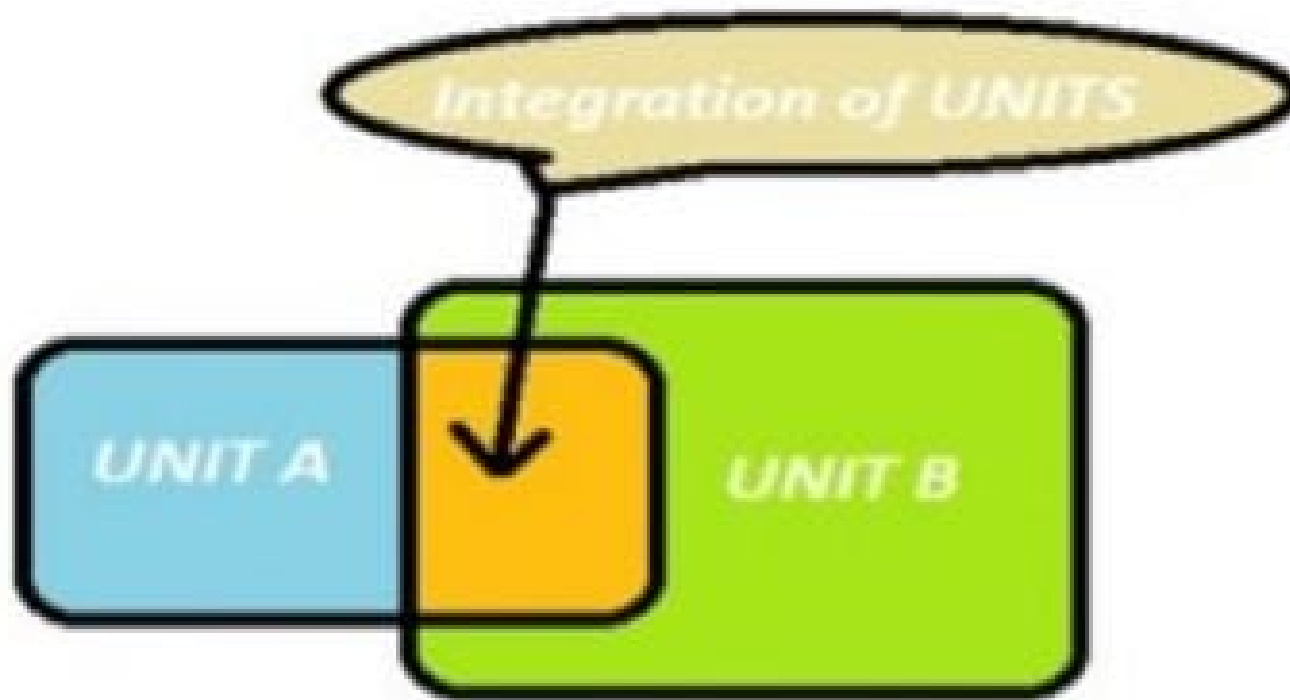


Functional Testing

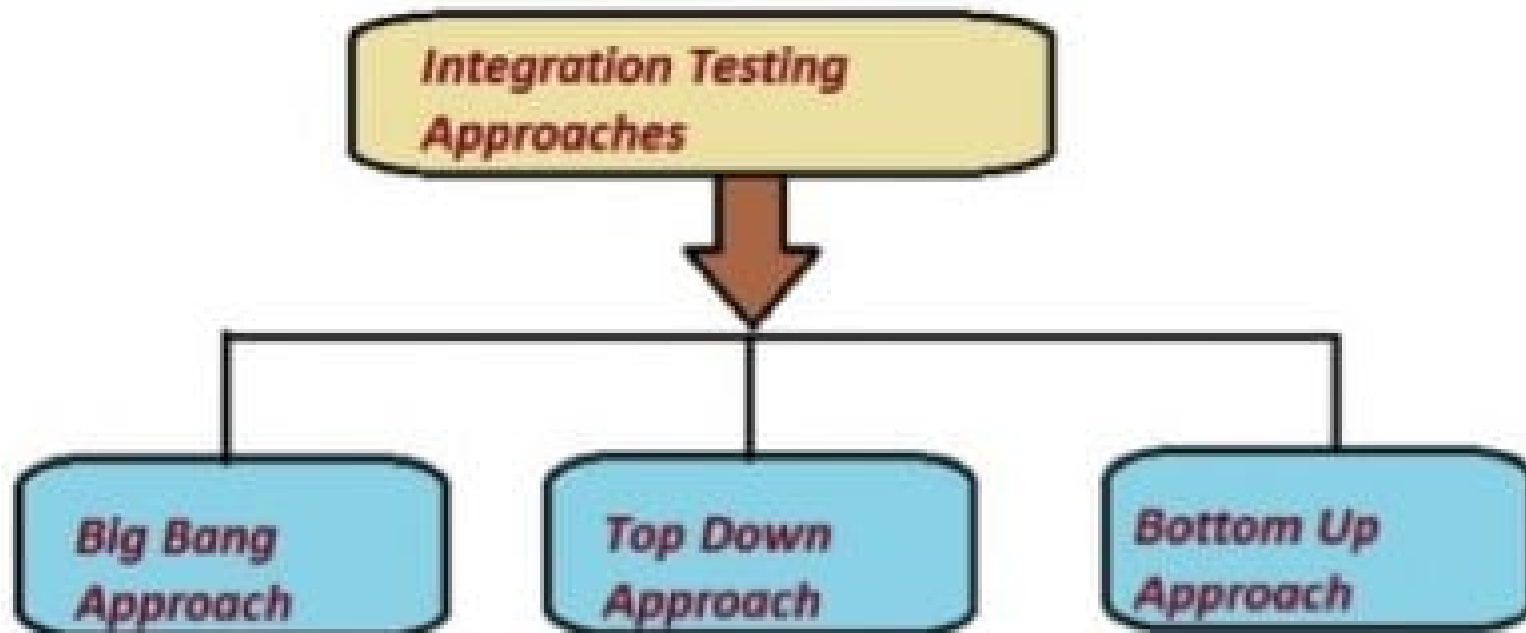
- Functional testing is a type of testing which verifies that each function of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing
- In layman terms, when you conduct functional testing you are interested in what the system can do

Integration Testing

- Integration testing is testing the integration of different part of the system together. Two different parts or modules of the system are first integrated and then integration testing is performed.

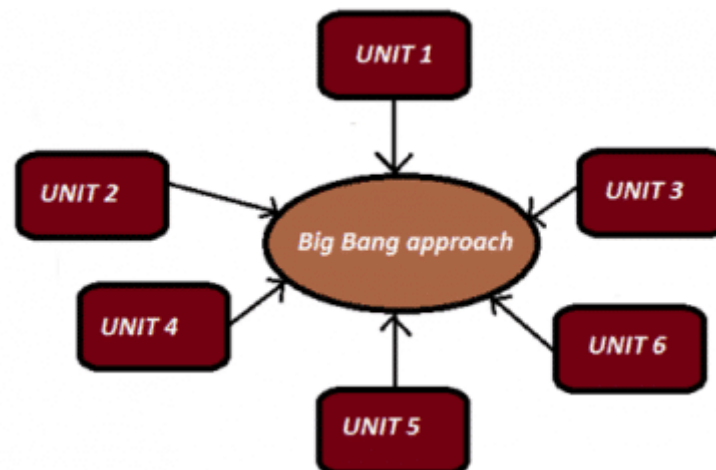


- There are 3 different types of Integration testing approaches



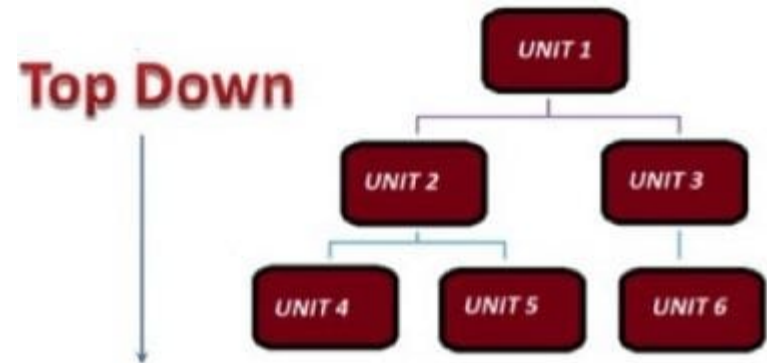
Big Bang Integration Approach

- In this approach, all the modules or units are integrated and tested as a whole at one time.
- This is usually done when the entire system is ready for integration testing at a single point of time.
- The big bang approach's major advantage is that everything integrated is tested at one time.
- One major disadvantage is that it becomes difficult to identify the failures.



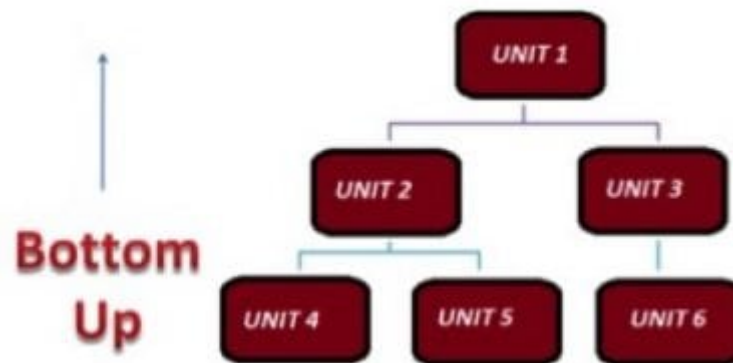
Top-Down Approach

- Integration of the units/modules is tested from the top to bottom levels step by step.
- The first unit is tested individually by writing test STUBS. After this, the lower levels are integrated one by one until the last level is put together and tested.
- Stub: Is called by the Module under Test.
- The only concern with this approach is that the major functionality is tested at the end.



Bottom-Up Approach

- Bottom-up Integration Testing is a strategy in which the lower level modules are tested first. These tested modules are then further used to facilitate the testing of higher level modules. The process continues until all modules at top level are tested
- Driver: Calls the Module to be tested.
- The major disadvantage of this approach is that the higher-level issues can only be identified at the end when all the units have been integrate

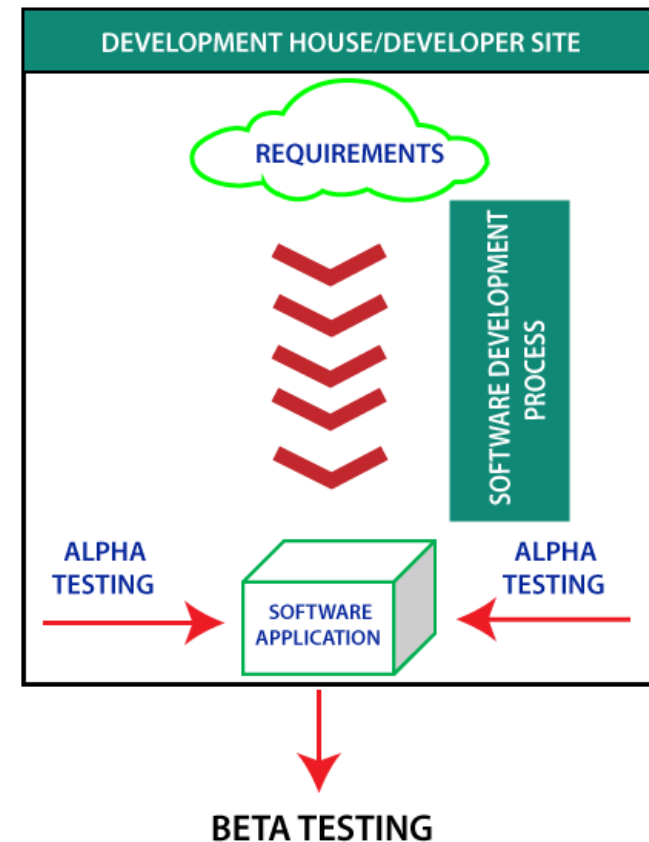


User Acceptance Testing

- UAT is to assess whether the Product is working for the user, correctly for the usage.
- This is also termed as End-User Testing
- The term “User” here signifies the end-users to whom the Product/application is intended and hence, testing is performed from the end-users perspective and from their point of view.
- These are the three types of UAT Testing
 - Alpha Testing
 - Beta Testing
 - Contract Acceptance Testing

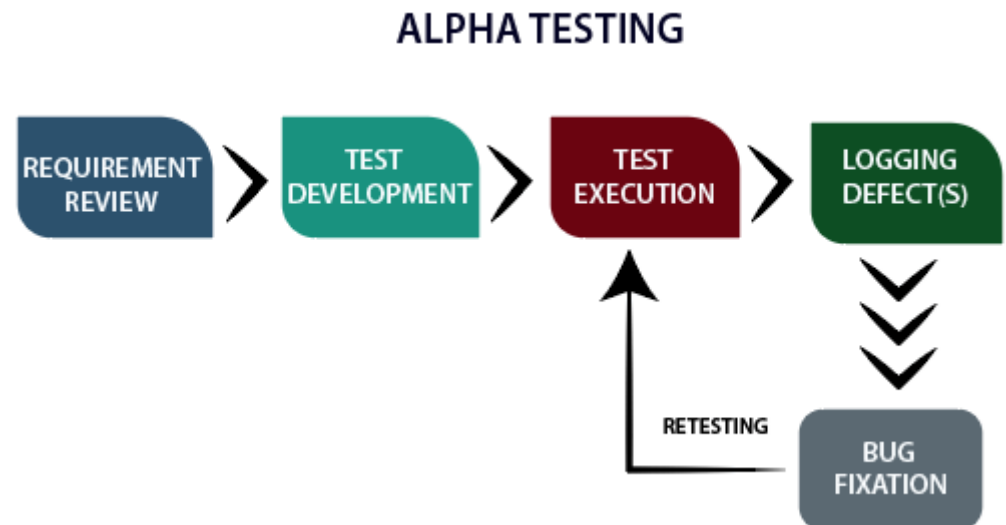
Alpha Testing

- This is to assess the Product in the development/testing environment by a **specialized testers team** usually called alpha testers.
- Testers feedback, suggestions help to improve the Product usage and also to fix certain bugs.
- Testing is happens at development site but not in client environment



Alpha Testing Process

- Alpha testing follows the following process
 - **Requirement Review:** Review the design of the specification and functional requirement
 - **Test Development:** Develop the test cases and test plan.
 - **Test case design:** Execute the test plan and test cases.
 - **Logging Defects**
 - **Bug Fixation**
 - **Retesting**



Advantage and Disadvantage

- Advantage

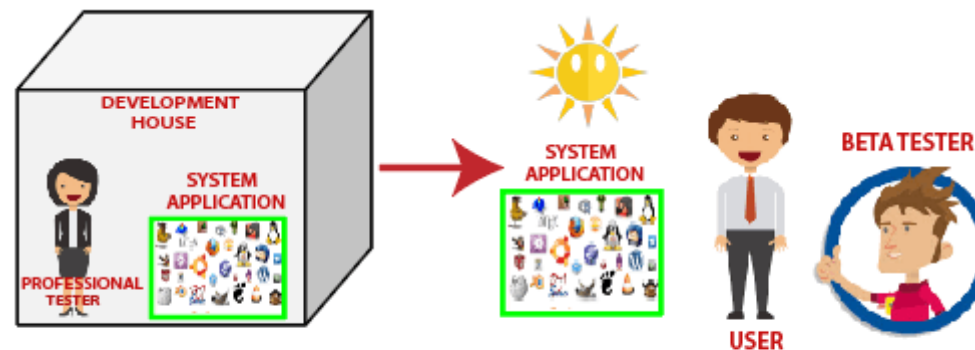
- Reduces the delivery time of the project.
- Every feedback helps to improve software quality.

- DisAdvantage

- The lab environment is used to simulate the real environment. But still, the lab cannot furnish all the requirement of the real environment
- The difference between the tester's tests the data for testing the software and the customer's data from their perspective may result in the discrepancy in the software functioning.

Beta Testing

- This is to assess the Product by exposing it to the **real end-users**, usually called beta testers/beta users, **in their environment**
- Continuous feedback from the users is collected and the issues are fixed
- Beta testing is the last phase of the testing, which is carried out at the client's or customer's site.



BETA TESTING OF THE PRODUCT IN
REAL WORLD ENVIRONMENT

Advantage and DisAdvantage

- Advantage

- Beta testing focuses on the customer's satisfaction.
- Beta testing helps to get direct feedback from users.
- It helps to reduce the risk of product failure via user validations.

- DisAdvantage

- No control over the process of the testing
- This testing can be a time-consuming process

Contract Acceptance Testing

- This is a contract which specifies that once the Product goes live, within a predetermined period, the acceptance test must be performed and it should pass all the acceptance use cases.
- Contract signed here is termed as Service Level Agreement (SLA), which includes the terms where the payment will be made only if the Product services are in-line with all the requirements, which means the contract is fulfilled.

Questions

- what is scrum
- What is Alpha testing
- What is beta testing
- Did you take part in any of the user acceptance testing
- Difference between Alpha and beta testing
- Advantages and disadvantages of Alpha and beta testing

- Who does the User Acceptance testing ?
- Difference between system and Integration testing
- Difference between sanity and smoke testing
- Can you explain roles in scrum ?
- Can you tell me duration of your scrum sprint
- What is user story point in scrum
- Tell me about the process of agile in your company ?
- What is the role of the scrum Master
- Explain the challenges you faced in Agile
-

- What is Stub ?
- What is driver ?
- What are type of integration testing
- Types of system testing
- Explain functional testing VS Non functional testing
- What is sprint backlog
- What is product backlog
-

- Responsibilities of scrum master
- What is the length of sprint or delivery duration ?
- Why V model is expensive ?
- What is critical functionality know as ?
- What is sanity testing
-

Non-Functional Testing

- Non-functional testing checks all the aspects not covered in functional tests. It includes the performance, usability, scalability, and reliability of the software.
- We conduct non-functional testing to make sure that the interests of the end-user are respected. The product will not enjoy success unless you manage to meet customer expectations. Non-functional testing is essential to add market value to the product.
- The results of non-functional testing are measured on the scale. If functional testing specifies what the program should do, non-functional tests describe how it should behave

Why Non – Functional

- Non-Functional testing checks the ability to work in an external environment.
- Non-functional testing gives detailed knowledge of product behavior and used technologies. It helps in reducing the risk of production and associated costs of the software.

Types of Non functional Testing

- Below are some of the most used non functional testing types
 - Performance Testing
 - Security Testing
 - Load Testing
 -

Advantage and DisAdvantage

- Advantage

- It provides a higher level of security
- It improves the performance of the system.
- Overall time consumption is less as compared to other testing processes.

- Disadvantage

- Every time the software is updated, non-functional tests are performed again.
- Due to software updates, people have to pay to re-examine the software; thus software becomes very expensive.

Security testing

- The parameter defines how a system is safeguarded against deliberate and sudden attacks from internal and external sources. This is tested via Security Testing.

-

-



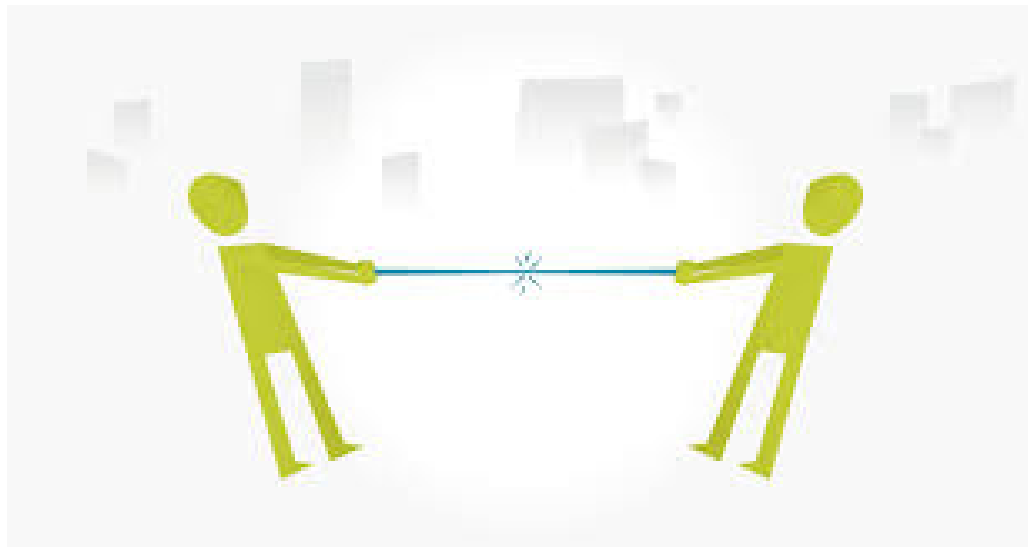
Performance testing

- performance testing is in general a testing practice performed to determine how a system performs in terms of responsiveness and stability under a particular workload



Stress

- Stress testing is a software testing activity that determines the robustness of software by testing beyond the limits of normal operation. Stress testing is particularly important for "mission critical" software, but is used for all types of software.

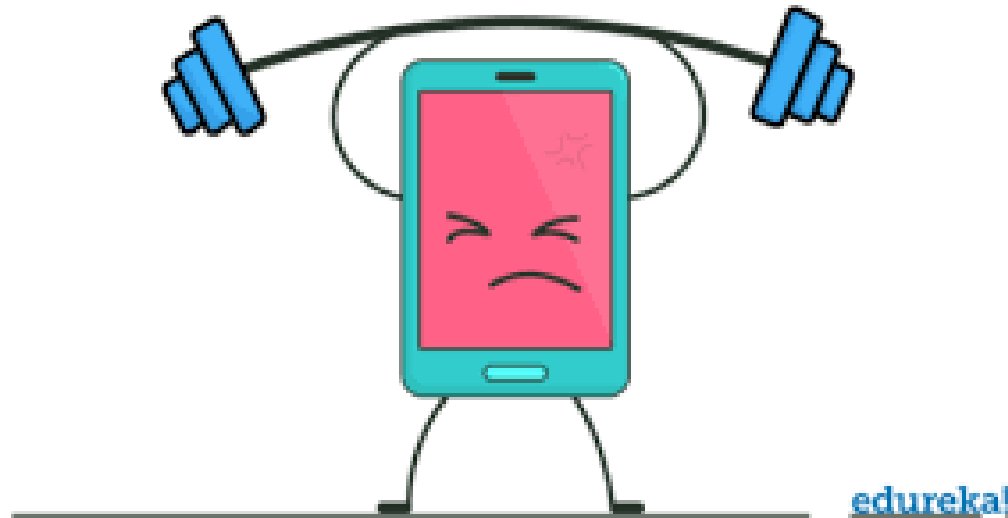


Load testing

- Load testing is the process of putting demand on a software system and measuring its response.

-

-



Accessibility testing

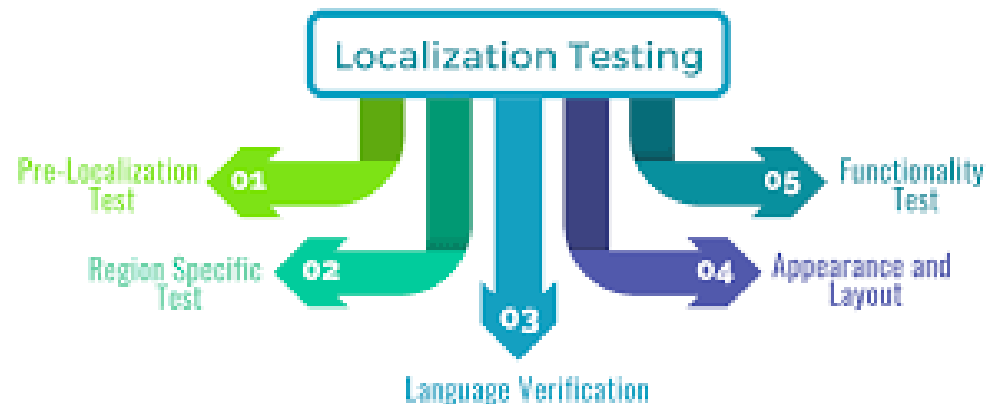
- Accessibility Testing is defined as a type of Software Testing performed to ensure that the application being tested is usable by people with disabilities like hearing, color blindness, old age and other disadvantaged groups

-
-
-



Localization Testing

- Localization testing is a software testing technique, which checks that an app or website offers full functionality and usability in a particular locale.
- If a product has been customized for a targeted language or region, localization testing verifies the accuracy and suitability of the content.



Internationalization Testing

- Internationalization testing is a non-functional testing technique.
- It is a process of designing a software application that can be adapted to various languages and regions without any changes.
- Localization, internationalization and globalization are highly interrelated
- .



Compatability testing

- Testing the application in a same environment but having different versions.

-



Usability testing

- The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system.
- Example – whats app

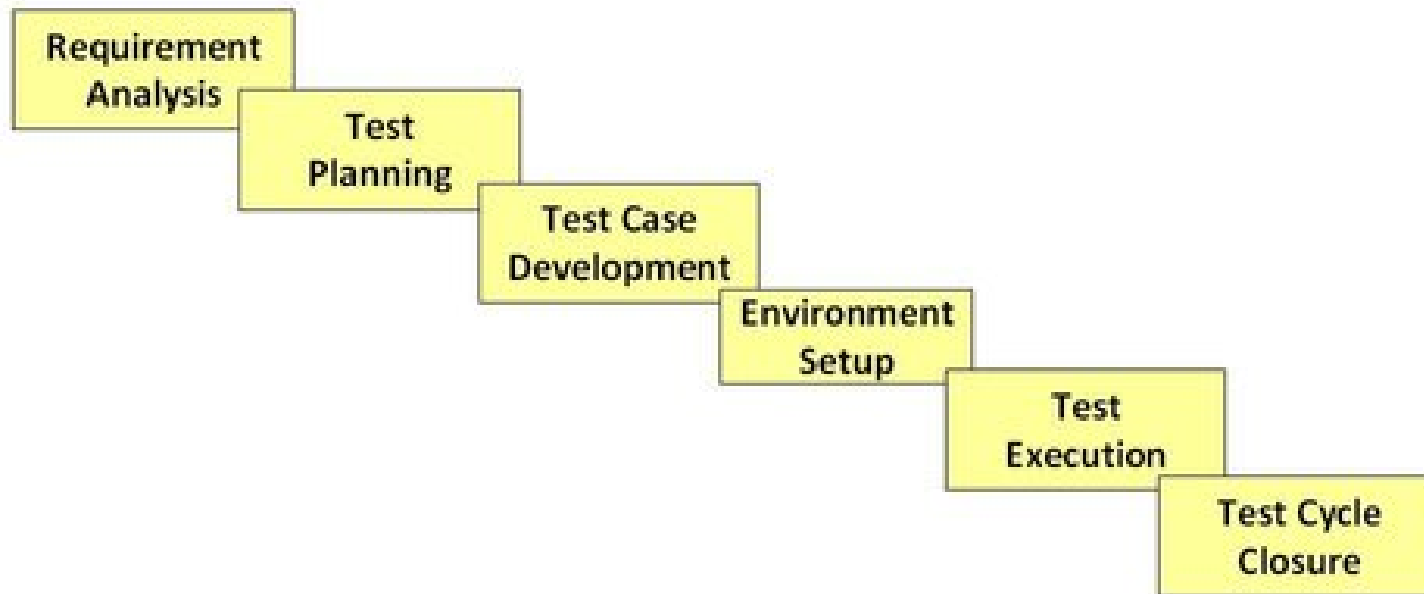


STLC

- STLC means software testing life cycle
- STLC is sequence of activity we perform during the testing process to ensure software quality goal is met
- STLC involves both verification and validation
- Software Testing is not just a single/isolate activity, i.e. testing. It consists of a series of activities carried out methodologically to help certify your software product.

STLC Phases

- Following six major phases in every Software Testing Life Cycle Model



Requirement Analysis

- Requirement Analysis in which test team studies the requirements from a testing point of view to identify testable requirements and the QA team may interact with various stakeholders to understand requirements in detail.
- Requirements could be either functional or non-functional.
- Automation feasibility for the testing project is also done in this stage.
- **Activities in Requirement Phase Testing**
 - Identify types of tests to be performed.
 - Gather details about testing priorities and focus.
 - Prepare Requirement Traceability Matrix (RTM).
 - Identify test environment details where testing is supposed to be carried out.
 - Automation feasibility analysis (if required).

Test Planning

- Senior QA manager determines the test plan strategy along with efforts and cost estimates for the project.
- the resources, test environment, test limitations and the testing schedule are also determined. The Test Plan gets prepared and finalized in the same phase.
- **Test Planning Activities**
 -
 - Preparation of test plan/strategy document for various types of testing
 - Test tool selection
 - Test effort estimation
 - Resource planning and determining roles and responsibilities.
 - Training requirement

Test Case Development

- The Test Case Development Phase involves the creation, verification and rework of test cases & test scripts after the test plan is ready
- Test Case Development Activities
 - Create test cases, automation scripts (if applicable)
 - Review and baseline test cases and scripts
 - Create test data (If Test Environment is available)

Test Environment Setup

- Test Environment Setup decides the software and hardware conditions under which a work product is tested.
- It can be done in parallel with the Test Case Development Phase
- **Test Environment Setup Activities**
 - Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
 - Setup test Environment and test data
 - Perform smoke test on the build

Test Execution Phase

- Test Execution Phase is carried out by the testers in which testing of the software build is done based on test plans and test cases prepared.
- The process consists of test script execution, test script maintenance and bug reporting
- **Test Execution Activities**
 - Execute tests as per plan
 - Document test results, and log defects for failed cases
 - Map defects to test cases in RTM
 - Retest the Defect fixes
 - Track the defects to closure

Test Cycle Closure

- Test Cycle Closure phase is completion of test execution which involves several activities like test completion reporting, collection of test completion matrices and test results
- **Test Cycle Closure Activities**
 - Evaluate cycle completion criteria based on Time, Test coverage, Cost, Software, Critical Business Objectives, Quality
 - Prepare test metrics based on the above parameters.
 - Document the learning out of the project
 - Prepare Test closure report.
 - Test result analysis to find out the defect distribution by type and severity.

SDLC VS STLC

Parameter	SDLC	STLC
Origin	Software Development Life Cycle	Software Testing Life Cycle
Objective	To complete successful development of the software	The only objective of STLC is testing
Requirement Gathering	BA gather the requirement and creates the Development plan	QA team analysis the functional and non-functional document and create test plan
High and Low level design	Developer creates the high and low level design plan	Test Analyst creates the integration test plan
Coding	Actual coding is take place	Testing team prepare the enviornment and execute them
Maintenance	Post Deployment support	Testers, execute regression suits, usually automation scripts to check maintenance code deployed.

Questionnaire on STLC

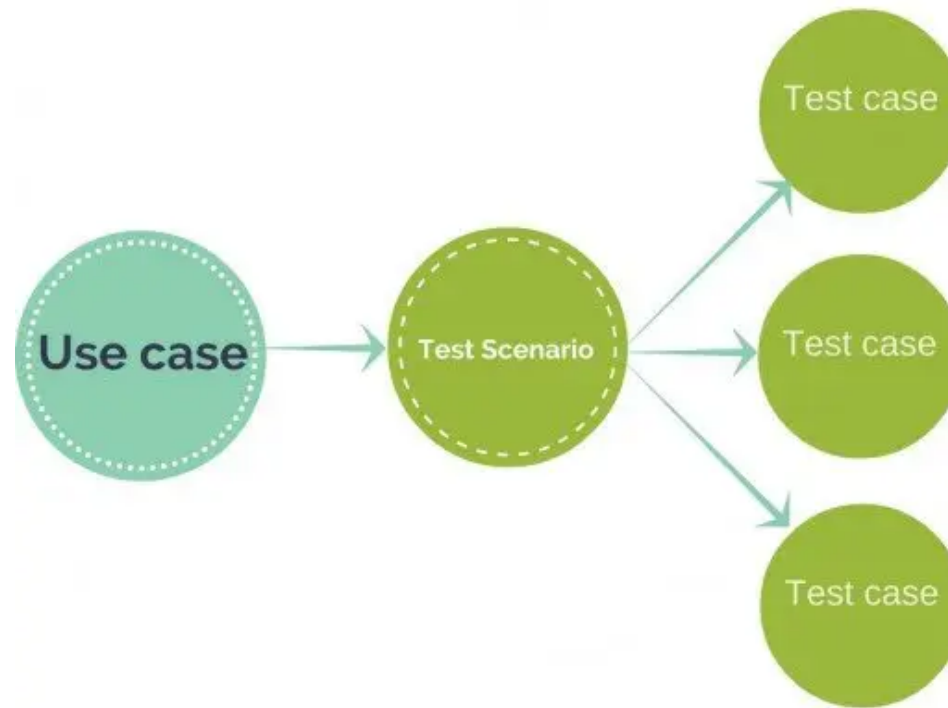
- What is a Software Testing Life Cycle (STLC)?
- What are the phases of STLC?
- What is the difference between SDLC and STLC?
- What activities are performed in each phase of STLC?
- What is the purpose of the test closure phase?
- How would you define that testing is sufficient and it's time to enter the Test Closure phase?
- Which phase of STLC includes bugs closure activities?

Test scenerios

- Test scenarios are mostly single line statement that tells what should be tested
- It cover end to end functionality of a software application
- Test scenarios just focus on “what to test”.
- Test scenarios are beneficial in quick testing of end to end functionality of the application.
- It saves a lot of time. Hence, these are better with projects having time constraints.

Test Scenario

- If client provided the use case then it breaks into test scenario which then breaks in test cases



Why create Test Scenarios?

- Creating Test Scenarios ensures complete Test Coverage
- Test Scenarios can be approved by various stakeholders like Business Analyst, Customers to ensure the Application Under Test is thoroughly tested. It ensures that the software is working for the most common use cases.
- They serve as a quick tool to determine the testing work effort and accordingly create a proposal for the client or organize the workforce.
- They help determine the most important end-to-end transactions or the real use of the software applications.
- For studying the end-to-end functioning of the program, Test Scenario is critical.

How to Write Test Scenarios

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

Test Application

- Verify the login functionality
- Verify the reset functionality
- Verify the cancel functionality

Test Application

User ID :	<input type="text"/>	
Password :	<input type="password"/>	
<input type="button" value="Login"/>	<input type="button" value="Cancel"/>	<input type="button" value="Reset"/>

Test Scenario Template and Assignment

- Template

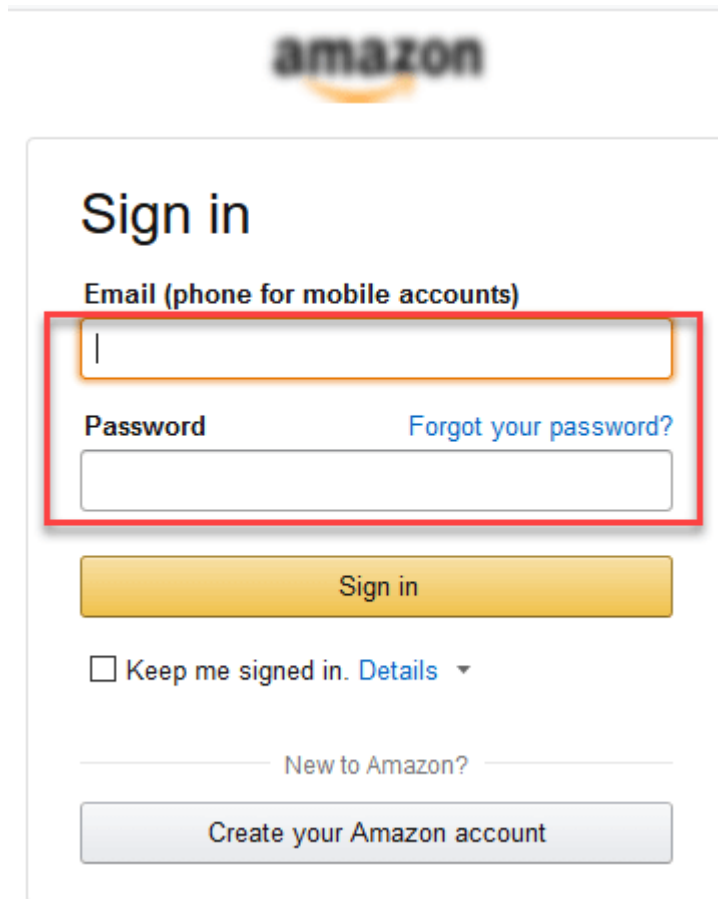
Test Scenario Template			
Module	RequirementId	TestScenarioId	Test Scenario Description
Login	[Optional]	TS_01	Verify that user is able to login with valid credentials.
	[Optional]	TS_02	Verify that user is not able to login with invalid credentials.
	[Optional]	TS_03	Verify mandatory field check validation on the login page.

- Assignment

- Write the test scenario for Amazon App/Whats App

Test Scenario for Login Page

- Verify the Login Functionality
- Verify create your amazon account functionality

The image shows the Amazon login page. At the top is the Amazon logo. Below it is a box titled "Sign in". Inside this box, there is a label "Email (phone for mobile accounts)" above a text input field. Below the input field is the label "Password" and a link "Forgot your password?". A red rectangle highlights the email input field and the password input field. Below the password field is a yellow "Sign in" button. Below the button is a checkbox labeled "Keep me signed in." with a link "Details" and a dropdown arrow. At the bottom of the box is a link "New to Amazon?" above a grey button labeled "Create your Amazon account".

amazon

Sign in

Email (phone for mobile accounts)

Password [Forgot your password?](#)

[Sign in](#)

☐ Keep me signed in. [Details](#) ▼

[New to Amazon?](#)

[Create your Amazon account](#)

Test Case

- A test case is a set of conditions for evaluating a particular feature of a software product.
- Once the test cases are created from the requirements, it is the job of the testers to execute those test cases

Test Case Design Technique

- Test Case design techniques help you design better test cases.
- Since exhaustive testing is not possible; Manual Testing Techniques help reduce the number of test cases to be executed while increasing test coverage
- It help to identify test conditions that are otherwise difficult to recognize.
- There are 5 different test case design techniues
 - Boundary value Analysis
 - Equivalence class Partitioning
 - Decision Table
 - State Trasition
 - Error Guessing

Boundary Value Analysis

- Boundary value analysis is based on testing the boundaries
- It includes maximum, minimum, inside or outside boundaries
- It gives a selection of test cases which exercise bounding values
- Example
 - There is input box which accept value between 1 to 10
 - Boundary values will be 0,1,2 and 9,10,11
- Guidelines for Boundary value Analysis
 - If any field takes value between values like X and Y then test cases should be designed with X-1, X,X+1 and Y-1,Y,Y+1
 - If any field takes value like greater than particular value then test case should consider maximum and minimum number
- **Assignment** – Covid vaccination is allowed between 18 to 60 years age people

Equivalence Class Partitioning

- In this technique, input test data for particular condition divides into each partition
- The test cases are then designed for each class or partition. This helps to reduce the number of test cases.
- **Example –**
 - Lets consider one field accept the value 1 to 10 and 20 to 30
 - It will have total 5 equivalence partition
 - Negative value to 0 (invalid)
 - 1 to 10 (valid)
 - 11 to 19 (invalid)
 - 20 to 30 (valid)
 - 31 to --- (invalid)
 - So instead of checking for the all values we will pick one value from each partitions like -3,5,11,21,32
- **Assignment –** Covid vaccination is allowed between 18 to 60 years age people

Decision Table

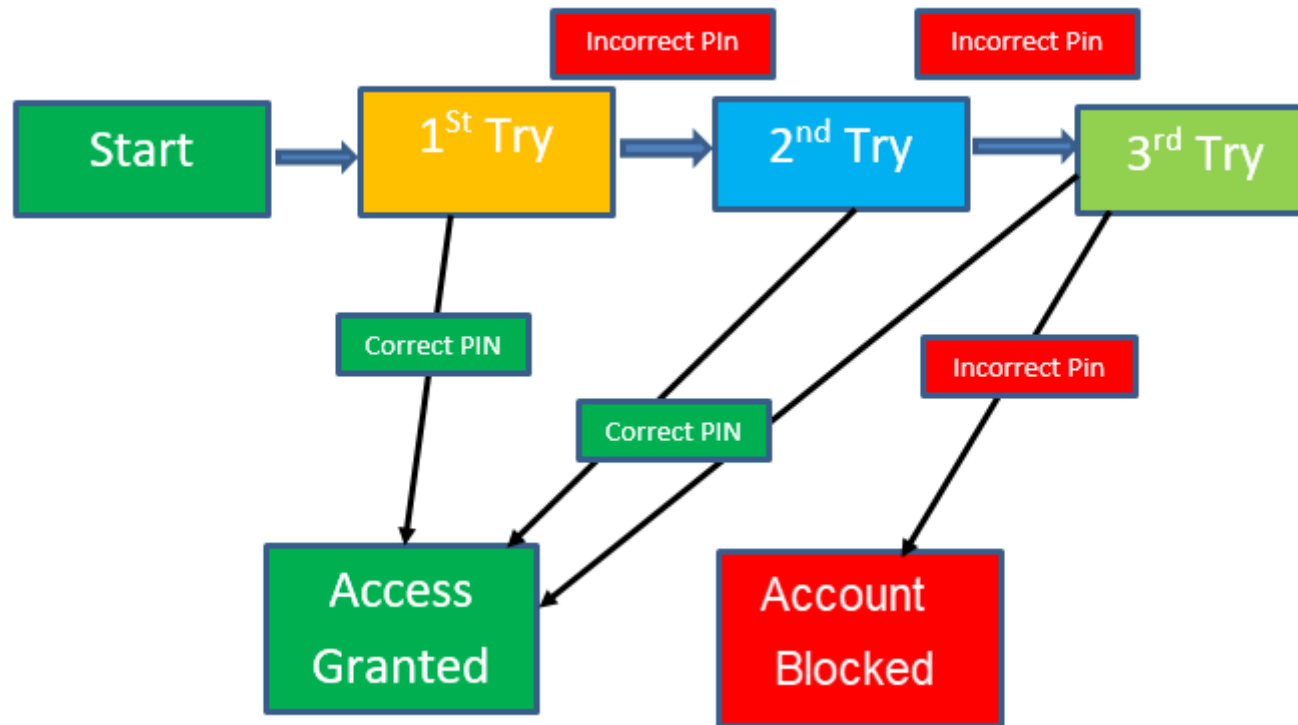
- It is also called as known as a cause-effect table.
- This approach is used when there is various input combinations and their respective system behavior are captured in a tabular form.
- Example
 - **Scenario** - ATM where a customer requests a cash withdrawal
 - **Business rule** -One of the business rules for the ATM is that the ATM machine pays out the amount if the customer has sufficient funds in their account or if the customer has the credit granted.

Conditions	R1	R2	R3
Withdrawal Amount <= Balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

Assignment – There is one user form having first name, last name, DOB, Education and submit button enables only all these details are filled

State Transition

- This technique is used where changes in input conditions change the state of the Application Under Test (AUT).
- It is used to capture the behavior of the software application when different input values are given to the same function.
- Example - if the user enters a valid password in any of the first three attempts the user will be able to log in successfully



State Transition

- when the user gives the correct PIN number, he or she is moved to Access granted state
- State transition table for above example

—

STATE	LOGIN	VALIDATION	REDIRECTED
S1	First Attempt	Invalid	S2
S2	Second Attempt	Invalid	S3
S3	Third Attempt	Invalid	S5
S4	Home Page		
S5	Error Page		

Error Guessing

- Error guessing is a technique in which there is no specific method for identifying the error
- It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software
- The main purpose of this technique is to identify common errors at any level of testing like
 - Enter blank space into the text fields.
 - Null pointer exception.
 - Enter invalid parameters.
 - Divide by zero.
 - Use maximum limit of files to be uploaded.
 - Check buttons without entering values.

Test Case Template

- This is the standard template we generally used
- It will be have following attributes
 - Test case Id
 - Module/Component
 - Priority
 - Test Case Description
 - Pre-requisite
 - Test Steps
 - Expected Result
 - Actual Result
 - Status
 - Executed by – Optional

Test Case Template									
TestCaselId	Component	Priority	Description/Test Summary	Pre-requisites	Test Steps	Expected Result	Actual Result	Status	Test Executed By
GoogleSearch_1	Search_Bar_Module	P0	Verify that when a user writes a search term and presses enter, search results should be displayed	Browser is launched	1. Write the url - http://google.com in the browser's URL bar and press enter. 2. Once google.com is launched, write the search term - "Apple" in the google search bar. 3. Press enter.	Search results related to 'apple' should be displayed	Search results with 'apple' keyword are displayed	Pass	TesterK

Test Case Vs Test Scenario

Test Case	Test Scenario
A test case contains clearly defined test steps for testing a feature of an application.	A test scenario contains a high level documentation, describing an end to end functionality to be tested.
Test cases focus on "what to test" and "how to test".	Test scenarios just focus on "what to test".
These have clearly defined step, pre-requisites, expected results etc. Hence, there is no ambiguity.	Test scenarios are generally one-liner. Hence, there is always possibility of ambiguity during testing.
Test cases can be derived from test scenarios. They have many to one relationship with the test scenarios.	These are derived from use cases.

Site for Testing

- <https://www.saucedemo.com/>

Severity and Priority

- Severity
 - It always relate with functionality.It represent how important that functionality is.
 - Generally it is categories into high,medium and low
- Priority
 - It always relate with the time. It represent how early it need to be fix
 - It also categories into high,medium and low

Examples of Severity and Priority

- If user is not able to login
- Application is developed by Indian company and it support chinesse language. There is some typo on home page due to which meaning will create comflict in two nations
- There is one module which support online payment and it is under development for application, Application is not able to connect to getways
- One application is having spelling mistake on terms and condition page , It should be we instead of I

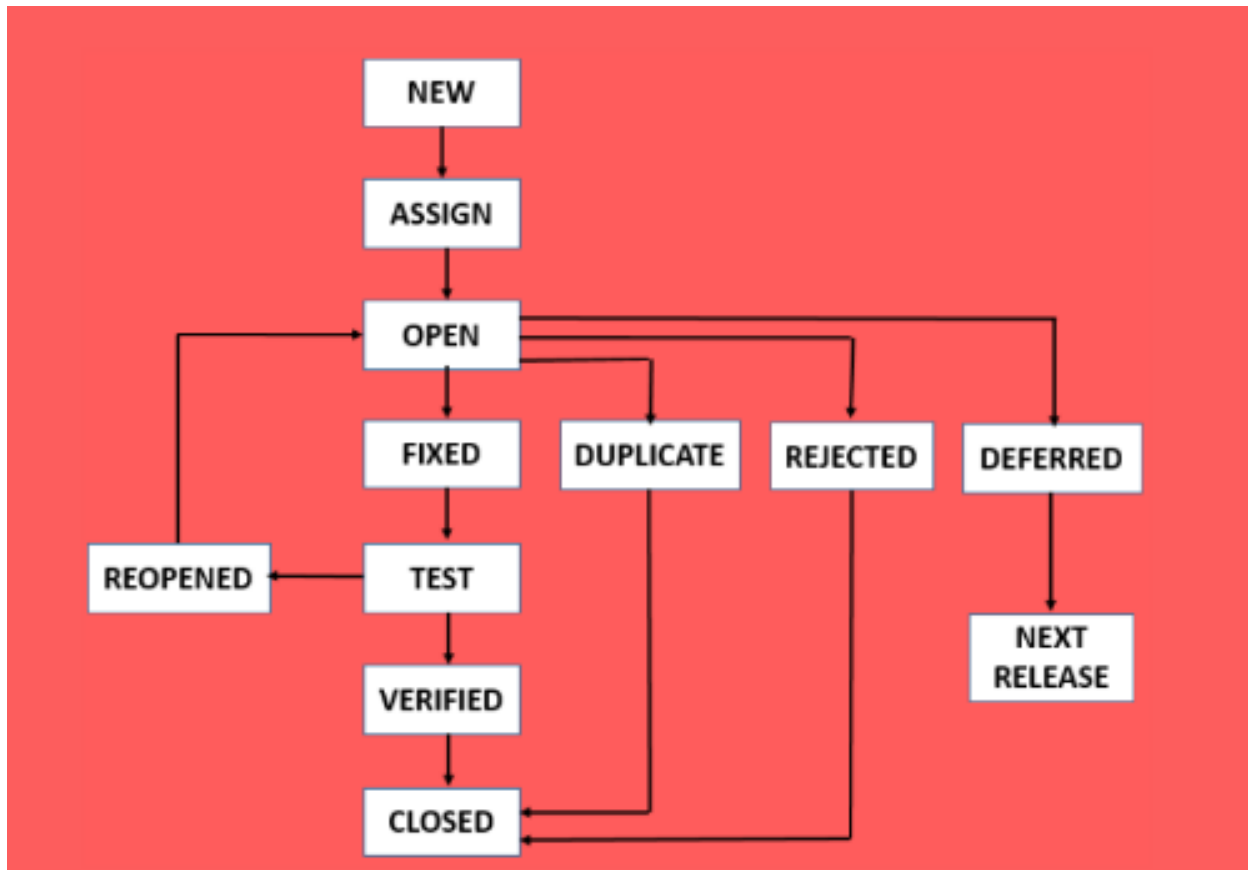
What is Defect

- A defect is an error or a bug, in the application which is created.
- A programmer while designing and building the software can make mistakes or error.
- These mistakes or errors mean that there are flaws in the software. These are called defects.
- A defect is a system error that doesn't allow the intended action to be completed
- It's important to start testing as early as possible because defects can be found throughout the entire software development process.
- The earlier we identify them, the lower the costs of their fixing.
- All bugs should be recorded and tracked so that they can be properly managed and resolved. Like it is tracked in excel, Jira and many more tools

Defect Template

[illegible]

Defect Life Cycle



Defect Example

- <https://www.saucedemo.com/>

QA and QC

- QA involves creating standards and processes to create a safe, effective process
- QC activities validate the product
- Neither QA or QC are optional. Similarly, it's impossible to say whether QA or QC is more valuable

Difference between them

QA	QC
QA aims to prevent the defect	QC aims to identify and fix defects
It is verification	It is validation
It does not involve executing the program	It always involves executing a program
It's a Proactive measure	It's a Reactive measure
process checklists, process standards, process documentation and project audit.	inspection, deliverable peer reviews and the software testing

Important terms

- Adoc testing
- Ad hoc Testing is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design techniques to create test cases.

Monkey testing

- Monkey testing is a type of random testing and no test cases used in this testing.

Exploratory

- Exploratory Testing is a type of software testing where Test cases are not created in advance but testers check system on the fly.

Done(DOD)

- The definition of done (DoD) is when all conditions, or acceptance criteria, that a software product must satisfy are met and ready to be accepted by a user, customer, team, or consuming system. We must meet the definition of done to ensure quality.

Ready

- Ready means that stories must be immediately actionable. The Team must be able to determine what needs to be done and the amount of work required to complete the User Story

Positive testing and Negative testing

- Negative Testing: Also known as “test to fail” – testing method where the tests’ aim is showing that a component or system does not work.
- Positive testing -Positive testing is a type of software testing that is performed by assuming everything will be as expected.

Spike testing

- Spike testing is a type of performance testing in which an application receives a sudden and extreme increase or decrease in load. The goal of spike testing is to determine the behavior of a software application when it receives extreme variations in traffic.

Pair testing

- Pair Testing: Software development technique in which two team members work together at one keyboard to test the software application. One does the testing and the other analyzes or reviews the testing.

Error Handling

- Error-Handling Testing:
- Software testing type which determines the ability of the system to properly process erroneous transactions.
-
-

Gorilla testing

- Software testing technique which focuses on heavily testing of one particular module

Mobile testing

- Mobile application testing is a process by which application software developed for handheld mobile devices is tested for its functionality, usability and consistency. Mobile application testing can be an automated or manual type of testing.

Types of mobile App

- Native Mobile Apps: Native mobile apps are designed to be “native” to one platform, whether it's Apple iOS, Google's Android, or Windows Phone. ...
- Hybrid Mobile Apps: These apps can be installed on devices just like native apps, but they run through web browsers. ...
- Web Apps:

Native Apps

- Hybrid Mobile Apps: These apps can be installed on devices just like native apps, but they run through web browsers.

Hybrid App

- Hybrid Mobile Apps: These apps can be installed on devices just like native apps, but they run through web browsers. ...

Web App

- A web application is application software that runs on a web server, unlike computer-based software programs that are run locally on the operating system of the device. Web applications are accessed by the user through a web browser with an active network connection

Types of mobile testing

- Accessibility testing
- Compatibility testing
- Version
- Devices
- Operating system
- UI
- Usability
- Networking
- Uninstallation /Installation testing

Emulator

- An emulator is a software program that allows your mobile to imitate the features of another computer or mobile software you want them to imitate by installing them to your computer or Mobile.

Emulator / Simulator Testing



Simulator

- A simulator is a program or machine that simulates a real-life situation, meaning that it creates a virtual version of it, often for the purpose of instruction or experiment, such as a flight simulator.

Difference between stimulator and Emulator

Stimulator	Emulator
Simulator's objective is to simulate the internal state of an object as close as possible to the internal state of an object.	The emulator aims at emulating or mimicking as close as possible the outer behavior of an object
Simulators are preferable whenever the testing team needs to test the mobile's internal behavior like its internal hardware, firmware, and so forth.	Emulators are preferable whenever the testing team needs to test the mobile's external behavior like calculating, making transactions, and so forth.
Simulators are written in high-level languages.	Emulators are written in machine-level assembly languages.
The simulators can be difficult in terms of debugging purpose.	Emulators are more suitable when it comes to debugging purpose
A simulator is just a partial re-implementation of the original software.	Often an emulator comes as a complete re-implementation of the original software

Types of Review

- Software Peer Reviews
- Software Management Reviews
- Software Audit Reviews

Software peer review

- Code review - The source code of the software is examined here. The software is checked for the bugs and the bugs are removed from the code.
- Pair programming: It is also a type of code review involving two people. The two people develop a code together at the same workstation.
- Inspection: It is a formal type of review where a person has to go through a defined set of instructions in order to find defect/defects. There can be a number of reviewers involved in this type of reviewing.
- Walkthrough: This is a process where the authors of the software as well as other associates are gathered at one place and they discuss about the software defects. Questions are made, comments are given, answers are given to all the queries people have regarding the software. With all the members' satisfaction, conclusions are made.
- Technical review: This is the team of qualified personnel examines the suitability of the software product for its intended discrepancies from the specification and standards point of view.
-

•Software Management Reviews

- :
- The management representatives are responsible for this type of review. The status of the work is evaluated and the decision by the activities of the software. This review is very important in making a decision regarding software

Software Audit Reviews:

-
- These are conducted by the personnel outside of the software project. They evaluate the software with specifications, standards, and other criteria.
-
- All the reviews are important to make software run successfully. Formal code reviews require a huge amount of investment of the time and the preparation of the review. Internal disputes also may carry leverage on these reviews. It is a time taking process and it may not be that accurate.
-
- There are certain guidelines that a person has to follow to perform a perfect review on build software.
-
- Defect Prevention is the main goal of the software as well as look out for the functionality of the software.
- Review of the requirement specifications should be done carefully so as to evaluate the software as per required.
- The list should be clarified.
- Correctness, portability, security and maintainability should be checked in build software.

Test Coverage

- What is Test Coverage?
 - It measures the amount of testing performed by a set of test
 - it is a technique to ensure that your tests are testing your code or how much of your code you exercised by running the test.
- Why we need it ?
 - To identify which requirement is not covered by test cases
 - To identify additional test cases to increase coverage
- What is the benefit?
 - Prevent Defect Leakage
 - It assure quality
 - Time, scope and cost can be kept under control

Test Coverage

- Formula to calculate Test Coverage
 1. The total lines of code in the piece of software quality you are testing
 2. The number of lines of code all test cases currently execute

If the number of lines of code in a system component is 500 and the number of lines executed across all existing test cases is 50, then your test coverage is:

$$(50 / 500) * 100 = 10\%$$

- Drawbacks
 - No tools to automate. Therefore, it takes lots of effort to analyze the requirements and create test cases.
 - There is always space for judgment errors.

Code Coverage

- What is code coverage?
 - It is one form of white box testing which finds the areas of the program not exercised by a set of test cases
- Why we need it ?
 - It defines the degree to which the source code has been tested
 - It helps to measure the efficiency of test implementation

Code Coverage Methods

- Following are major code coverage methods
 - Statement Coverage
 - Decision Coverage
 - Branch Coverage
 - Path Coverage

Statement Coverage

- **What is Statement?**

- Each line in source code is consider as statement

- **Example**

```
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. }
```

What is the total number of statements in above code?

Scenarios

If a=3 and b=5

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. }

**Total number of executed
statement =5**

If a=3 and b=-9

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. }

**Total number of executed
statement =6**

Statement Coverage

- Statement Coverage=
- $(\text{Total number of executed statement} / \text{Total statement}) * 100$
 - Scenario 1 = $(5/7) * 100 = 71\%$
 - Scenario 2 = $(6/7) * 100 = 85\%$
- Objective of it to make sure all the executable statements in the source code are executed at least once

Decision Coverage

- **What is Decision?**

- Each condition in source code which results in true or false outcomes

- **Example**

```
1. Demo(int a) {  
2.   If (a > 5)  
3.     a = a * 3  
4.   Print (a)  
5. }
```

What is the total number of decision **outcomes in above code?**

Scenarios

Value of a =2

```
Demo(int a) {  
    If (a> 5)  
  
    a=a*3  
  
    Print (a)  
  
}
```

**Total number of
executed decision
=1**

Value of a is 9

```
Demo(int a) {  
    If (a> 5)  
  
    a=a*3  
  
    Print (a)  
  
}
```

**Total number of
executed decision
=1**

Decision Coverage

- Decision Coverage=
- $(\text{Total number of executed decision} / \text{Total number of decision outcomes}) * 100$
 - Scenario 1 = $(1/2) * 100 = 50\%$
 - Scenario 2 = $(1/2) * 100 = 50\%$
- Objective of it to make sure all the executable decision in the source code are executed at least once

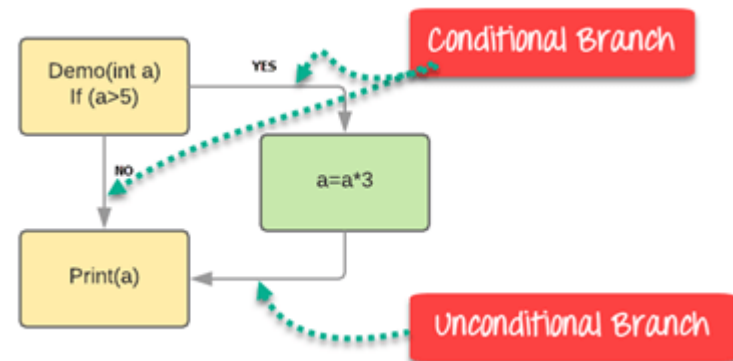
Branch Coverage

- **What is Branch?**

- Branch is nothing but a different execution flow occurs at decision point

- **Example**

```
1. Demo(int a) {  
2.   If (a > 5) {  
3.     a = a * 3  
4.   } Print (a)  
5. }
```



What is the total number of branches in above code?

Scenarios

Value of a =2

```
Demo(int a) {  
    If (a> 5)  
  
    a=a*3  
  
    Print (a)  
}
```

**Total number of
executed branches
=1**

Value of a is 9

```
Demo(int a) {  
    If (a> 5)  
  
    a=a*3  
  
    Print (a)  
}
```

**Total number of
executed branches
=2**

Branch Coverage

- Branch Coverage=
$$\left(\frac{\text{Total number of executed branches}}{\text{Total number of branches}} \right) * 100$$
- Scenario 1= $(1/3)*100=33\%$
- Scenario 2= $(2/3)*100=67\%$
- Objective of it to make sure all the executable branch in the source code are executed atleast once
- Key Point
 - There is no major difference between decision and branch coverage , only difference is branch coverage will consider **unconditional branches** as well

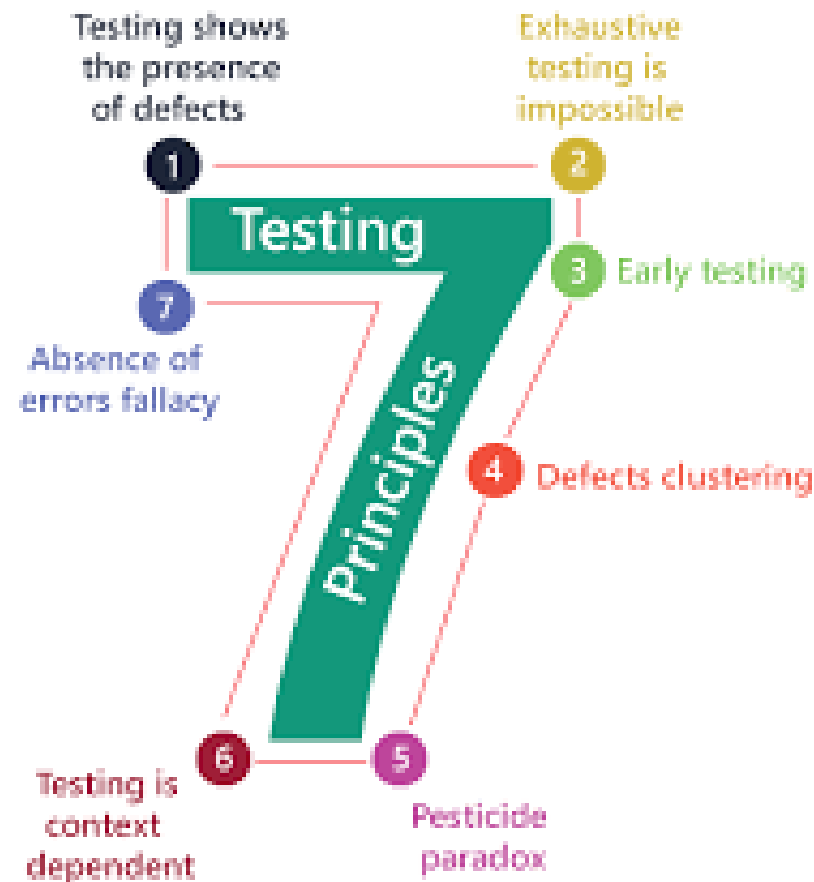
Condition Coverage

- What is condition coverage?
 - Condition coverage is to check individual outcomes for each logical condition
- **Example**
 - If ((x<y) && (a>b)) {
 syso("");
 - }
 - Y is greater than x – T a is greater than b-T
 - TF
 - FT
 - FF

Point to Remember

- 100% Branch/Decision coverage will imply 100% Statement coverage, **then whether it means we don't need to do statement coverage?**
- If you achieve 100% coverage it doesn't mean that your code is bug free

7 principal of testing



Testing shows presence of defect

- The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn't talk about the absence of defects. Software testing can ensure that defects are present but it can not prove that software is defect-free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not removes all defects.

Exhaustive testing is impossible

- It is the process of testing the functionality of the software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test case. It can test only some test cases and assume that the software is correct and it will produce the correct output in every test case. If the software will test every test case then it will take more cost, effort, etc., and which is impractical.

Early testing

- To find the defect in the software, early test activity shall be started. The defect detected in the early phases of SDLC will be very less expensive. For better performance of software, software testing will start at the initial phase i.e. testing will perform at the requirement analysis phase.

Defect clustering

- In a project, a small number of the module can contain most of the defects. Pareto Principle to software testing state that 80% of software defect comes from 20% of modules.
- Example – Banking Application is not working so the related module will be affected

Pesticide Paradox

- Repeating the same test cases, again and again, will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.
- Example- Insect killer

Testing is content dependent

- The testing approach depends on the context of the software developed. Different types of software need to perform different types of testing.
- For example, The testing of the e-commerce site is different from the testing of the Android application.

Absence of error Fallacy

- If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it is also mandatory to fulfill all the customer requirements.

-