



Session-8

Testing in the Software Lifecycle [SDLC]

(software development life cycle)

DR. BHARATI WUKKADADA

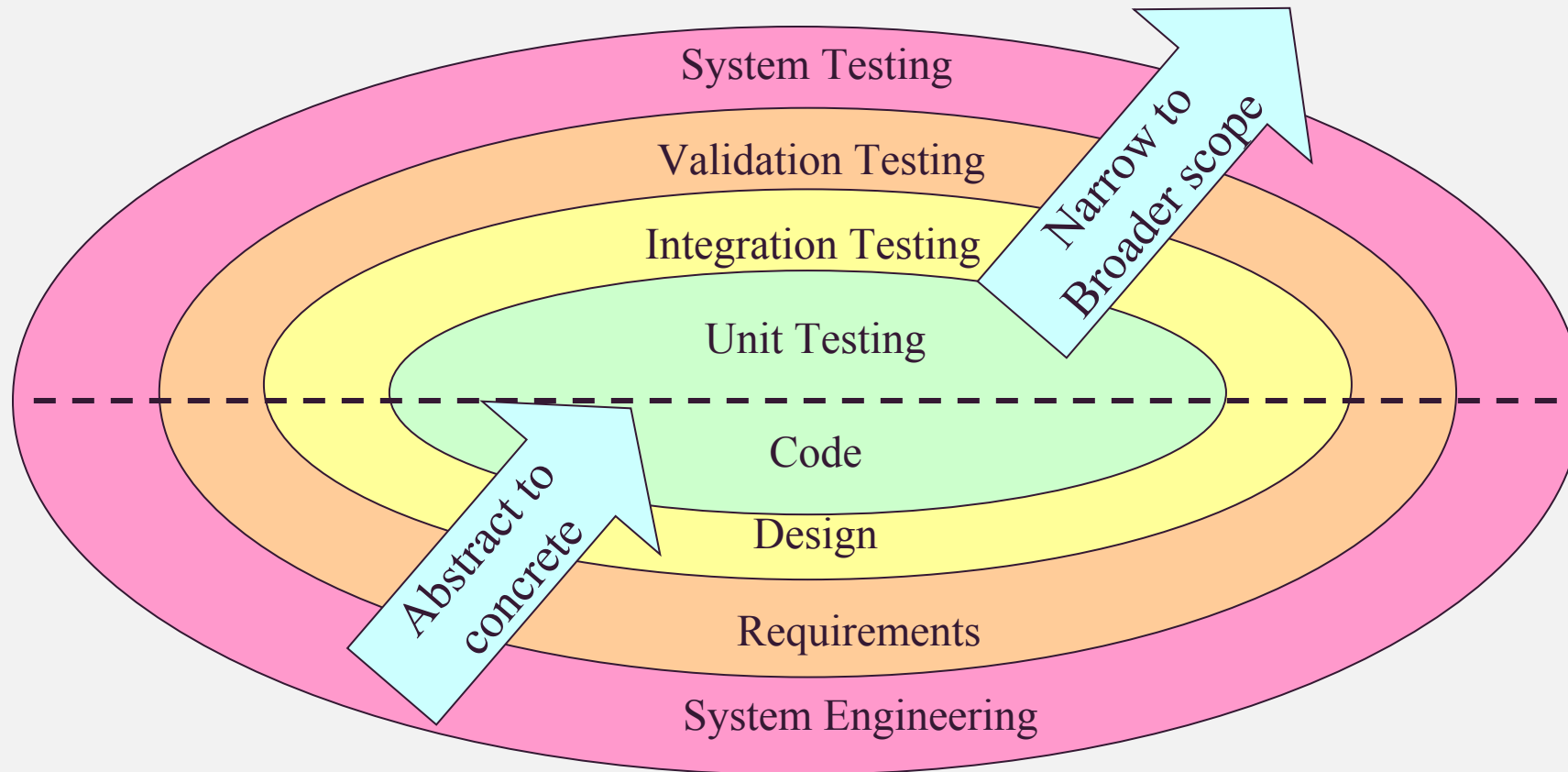
Introduction

- ◆ *SDLC stands for Software Development Lifecycle*
- ◆ SDLC is a framework defining tasks performed at each step in software development process.
- ◆ SDLC aims to produce a high-quality system that meets or exceeds customer expectations.





A Strategy for Testing Conventional Software



Content to be covered...

Testing in the Software Lifecycle

Content A

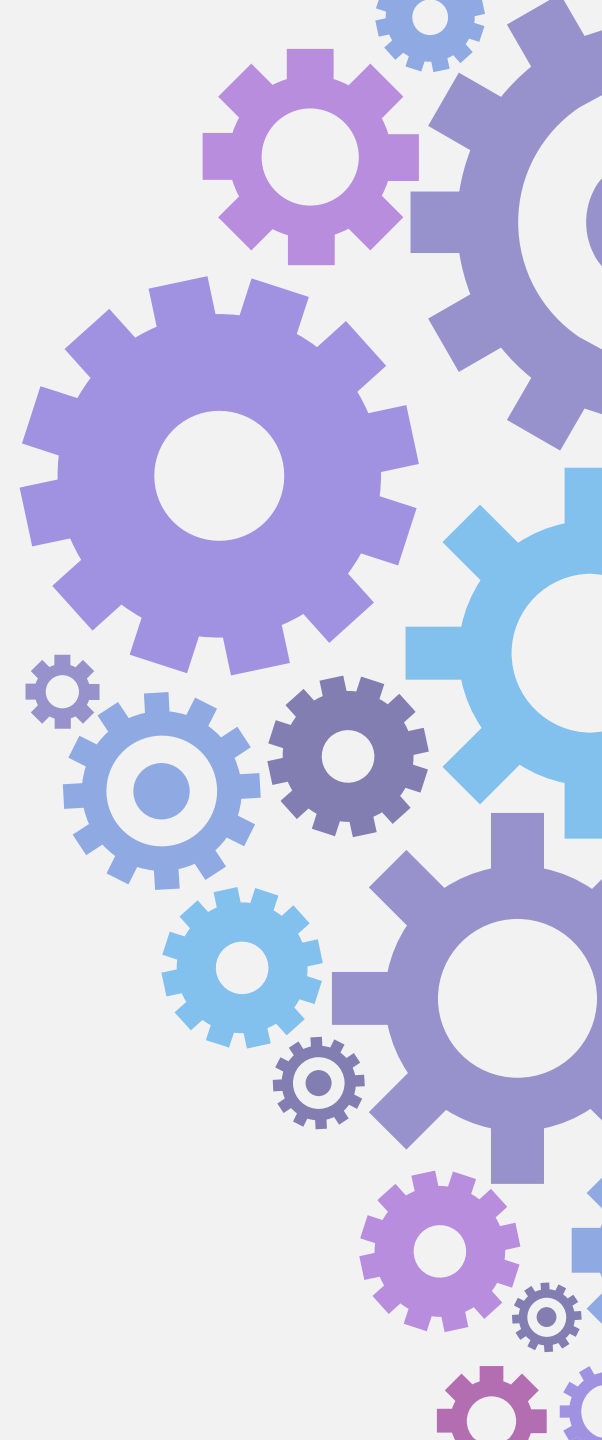
- *W Model*
- *V Model*

Content B

- *Functional Testing*
- *Non – Functional Testing*

Content C

- *Component Testing*
- *Integration Testing*
- *Acceptance Testing*
- *System Testing*





V Model

(The **V-model** is an SDLC model where execution of processes happens in a sequential manner in a V-shape.)

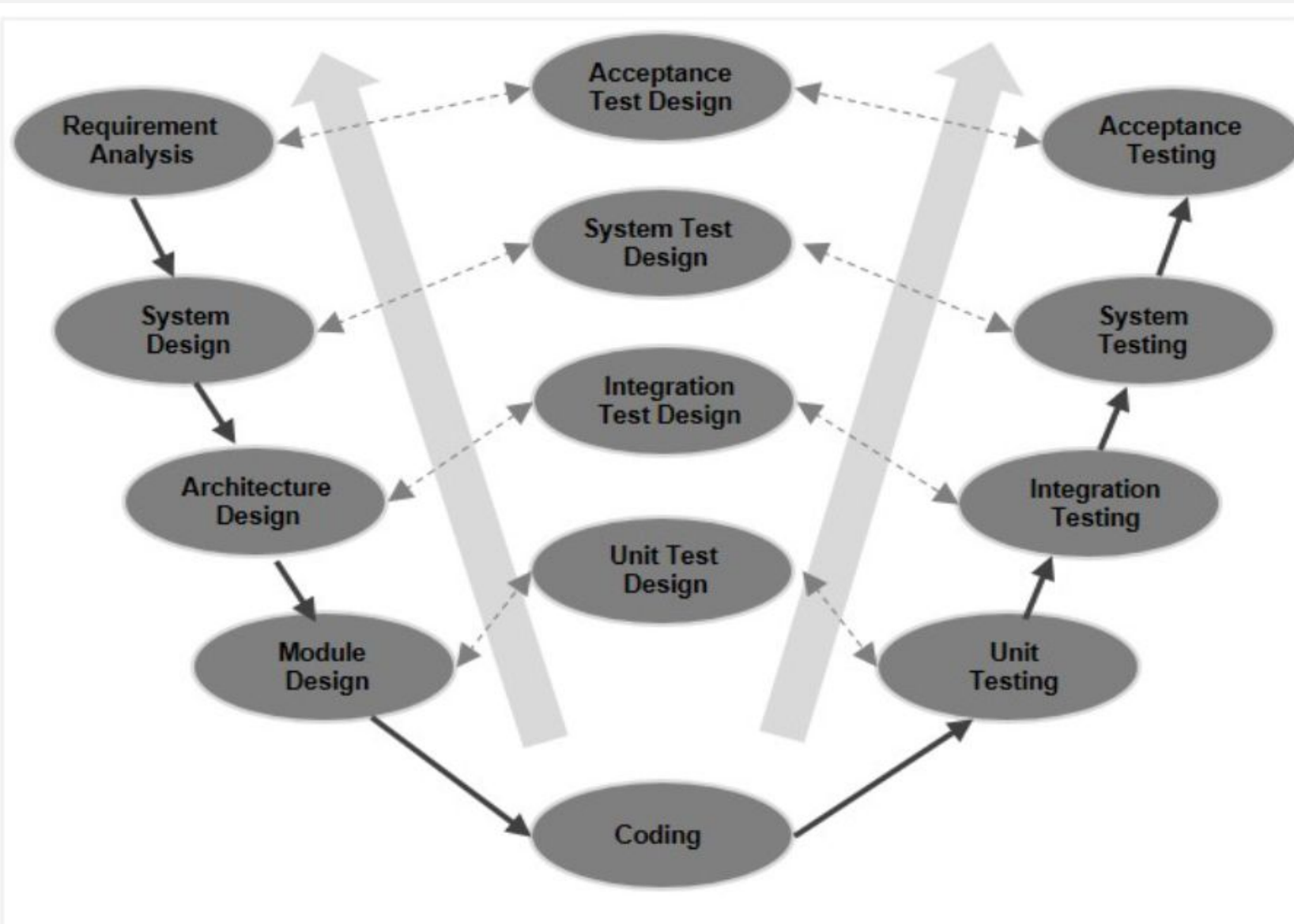


Introduction

- It is also known as **Verification and Validation model**.
- **V Model** is a highly disciplined SDLC model in which there is a testing phase parallel to each development phase.

Under the V-Model, the corresponding testing phase of the development phase is planned in parallel. So, there are Verification phases on one side of the 'V' and Validation phases on the other side. The Coding Phase joins the two sides of the V-Model.

The following illustration depicts the different phases in a V-Model of the SDLC.



Verification



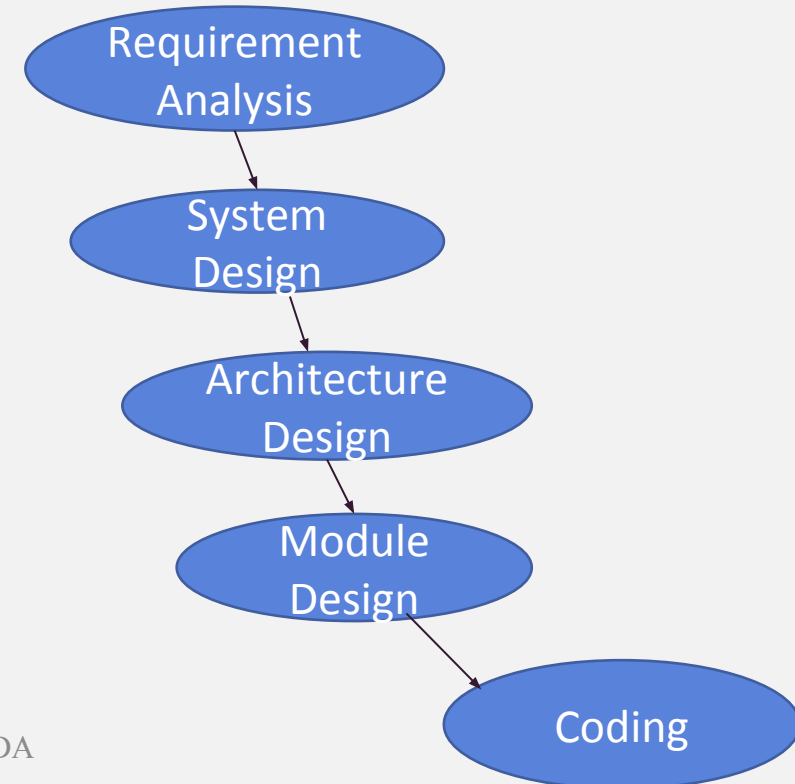
Validation

- Verification is **Static kind** of testing where we don't execute the source and test the application.
- Verification is the process of evaluating products of a development phase.
- Following items are evaluated during Verification: Plans, Requirement Specifications, Design Specifications, Code, etc.

- Validation is the **Dynamic kind** of testing where we execute the source code and do the application testing.
- Validation is the process of evaluating software at the end of the development process .
- Following item is evaluated during Validation: Actual product or Software under test.

Development Phases

- Development phase is also known as verification phase.
- There are the various phases of Verification Phase of V-model:
 - a.) Business requirement analysis
 - b.) System Design
 - c.) Architecture Design
 - d.) Module Design
 - e.) Coding Phase

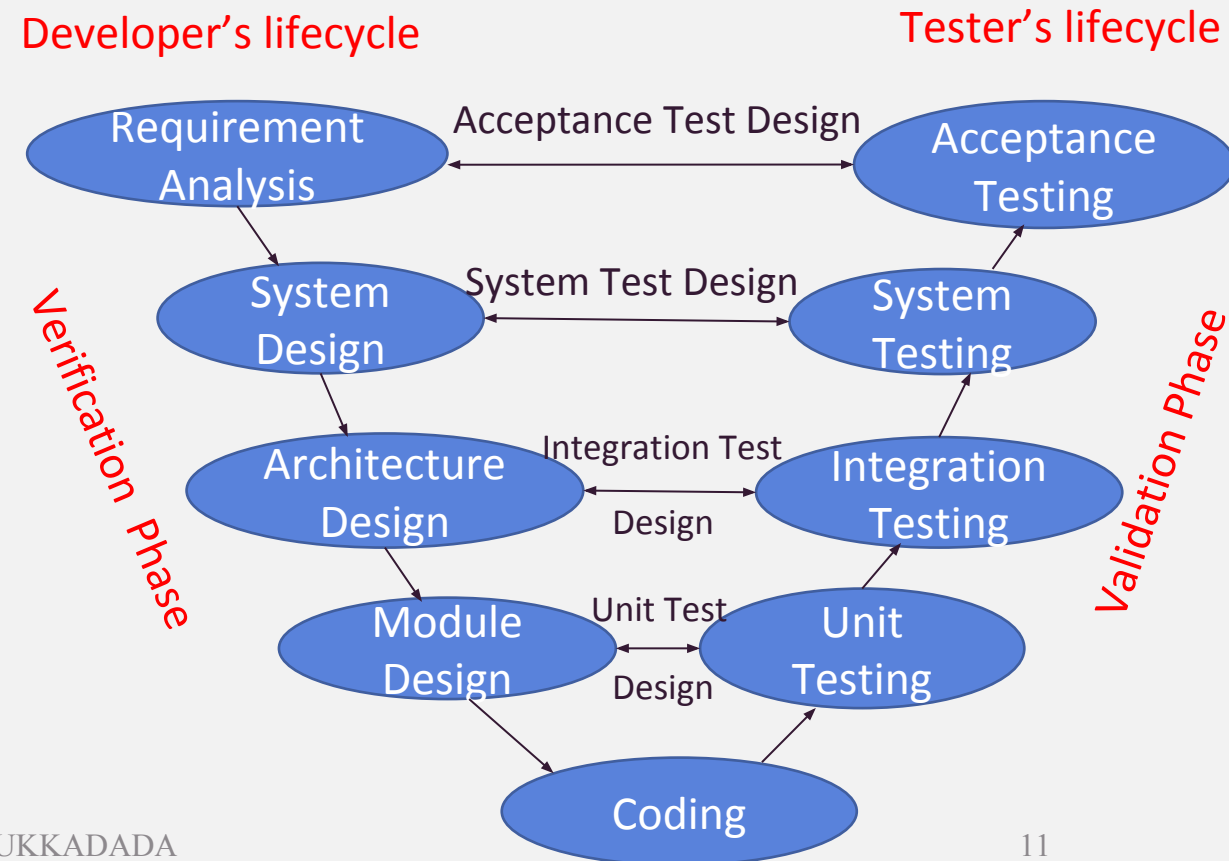


Test Stages

- Test Stages is also known as validation phase.
- There are the various phases of

Validation Phase of V-model:

- a.) Unit Testing
- b.) Integration Testing
- c.) System Testing
- d.) Acceptance Testing



- **Requirement Analysis:** This phase contains detailed communication with the customer to understand their requirements and expectations. This stage is known as Requirement Gathering.
- **System Design:** This phase contains the system design and the complete hardware and communication setup for developing product.
- **Architectural Design:** System design is broken down further into modules taking up different functionalities. The data transfer and communication between the internal modules and with the outside world (other systems) is clearly understood.
- **Module Design:** In this phase the system breaks down into small modules. The detailed design of modules is specified, also known as Low-Level Design (LLD).

- **Unit Testing:** Unit Test Plans are developed during module design phase. These Unit Test Plans are executed to eliminate bugs at code or unit level.
- **Integration testing:** After completion of unit testing Integration testing is performed. In integration testing, the modules are integrated and the system is tested. Integration testing is performed on the Architecture design phase. This test verifies the communication of modules among themselves.
- **System Testing:** System testing test the complete application with its functionality, inter dependency, and communication. It tests the functional and non-functional requirements of the developed application.
- **User Acceptance Testing (UAT):** UAT is performed in a user environment that resembles the production environment. UAT verifies that the delivered system meets user's requirement and system is ready for use in real world.

Principles of V - Model

- The principles of v model are as follows :

a.) Large to Small

b.) Data/Process Integrity

c.) Scalability

d.) Cross Referencing

e.) Tangible Documentation

- **Large to Small:** In V-Model, testing is done in a hierarchical perspective, For example, requirements identified by the project team, create High-Level Design, and Detailed Design phases of the project. As each of these phases is completed the requirements, they are defining become more and more refined and detailed.
- **Data/Process Integrity:** This principle states that the successful design of any project requires the incorporation and cohesion of both data and processes. Process elements must be identified at each and every requirements.
- **Scalability:** This principle states that the V-Model concept has the flexibility to accommodate any IT project irrespective of its size, complexity or duration.
- **Cross Referencing:** Direct correlation between requirements and corresponding testing activity is known as cross-referencing.
- **Tangible Documentation:** This principle states that every project needs to create a document. This documentation is required and applied by both the project development team and the support team. Documentation is used to maintaining the application once it is available in a production environment.

Industrial Challenge

- Accurately define and refine user requirements.
- Design and build an application according to the authorized user requirements.
- Validate that the application they had built adhered to the authorized business requirements.

Why preferred?

- Each phase of V-Model has specific deliverables and a review process.
- Proactive defect tracking – that is defects are found at early stage.
- The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed.

When to use...

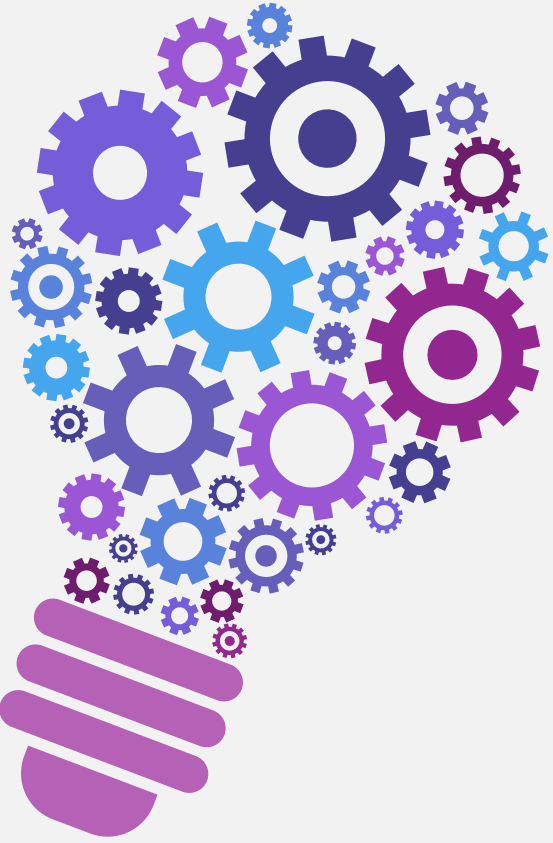
- The requirement is well defined and not ambiguous
- Acceptance criteria are well defined.
- Project is short to medium in size.
- Technology and tools used are not dynamic.

Advantages

- This is a highly-disciplined model and Phases are completed one at a time.
- Works well for smaller projects where requirements are very well understood.
- Simple and easy to understand and use.
- Suited for Restricted Projects.
- Ideal for Time Management.

Disadvantages

- Not a good model for complex and object-oriented projects.
- Not suitable for the projects where requirements are at a moderate to high risk of changing.
- Once an application is in the testing stage, it is difficult to go back and change a functionality.
- Lacks Adaptability.
- Suited for Lengthy Life Cycles.



W Model

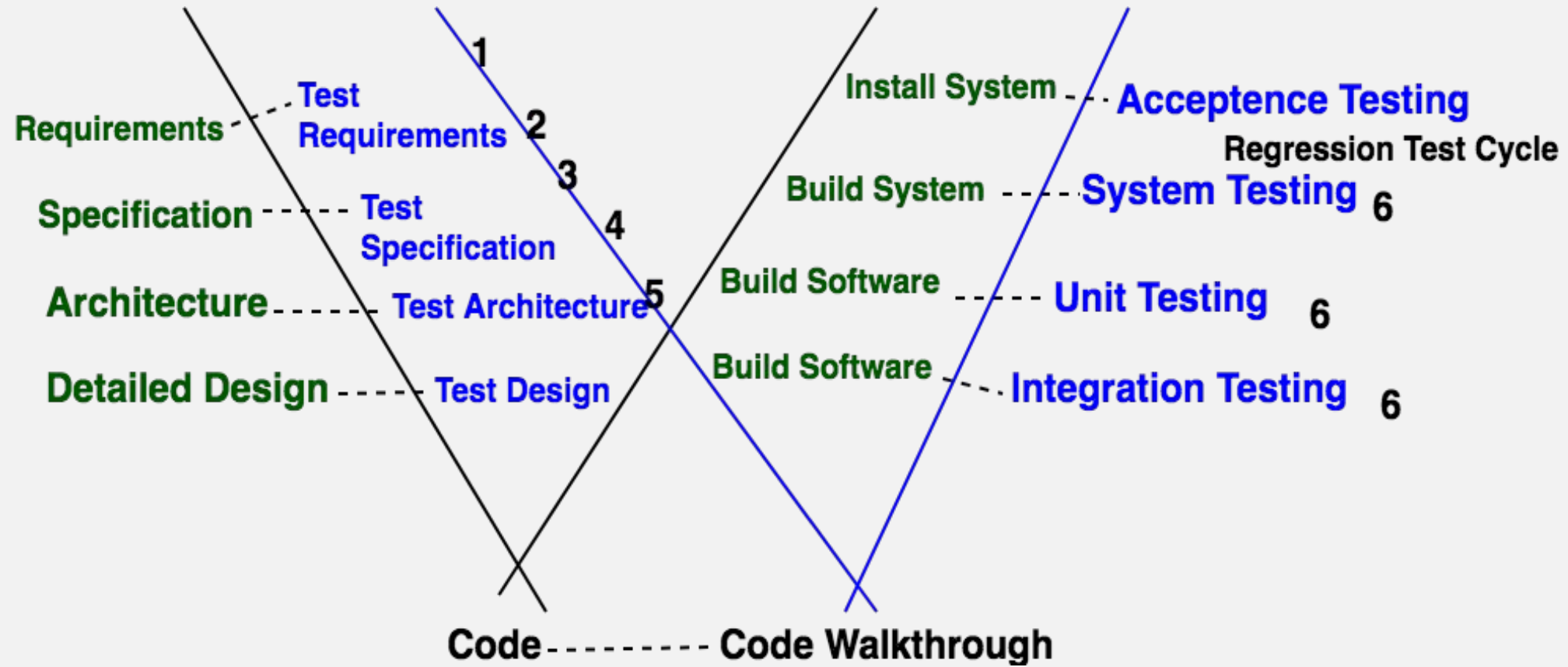


Introduction to W Model

- **W-model** is the most recent software development model where we start real testing activity simultaneously software development process starts.
- Where as software development process is a method in which a software or product is made through various stages of planning, development and testing before the final software or product is delivered.
- Testing is such a stage that is extremely crucial to ensure the delivery of an optimum quality product.

- W-Model covers those activities which are skipped by V-Model and also, it deals with problems which couldn't be catch by V-Model.
- W-Model approach attempts to address and tackle the shortcomings of V-Model.
- W-model can be done only once the development of the product is complete with no modifications required to be done in between. This type of testing **is most suitable for short projects**.
- With the help of W-Model, we ensure that the testing of the product starts from the very first day of the inception of product and each phase of the product development is verified and validated.

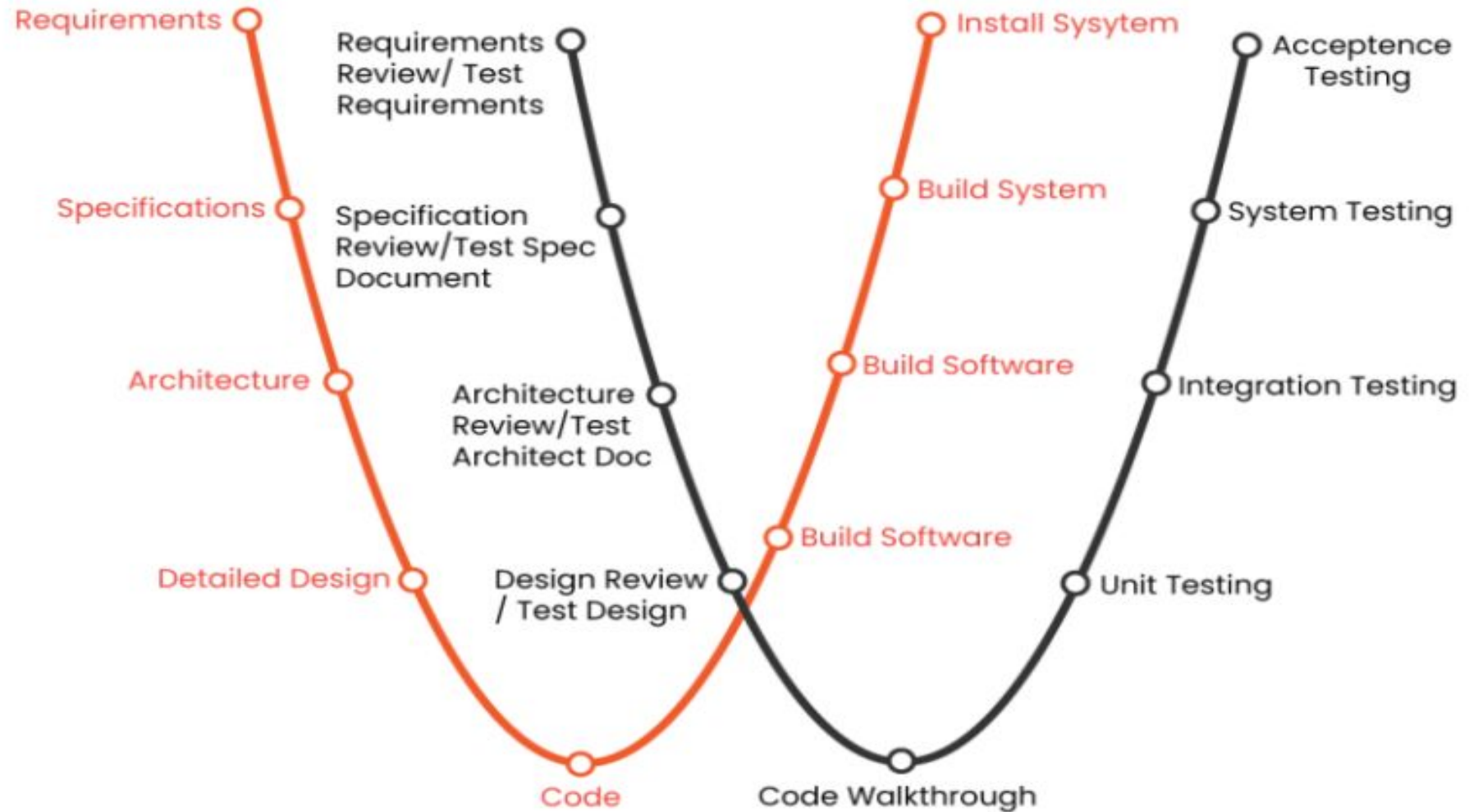
W Model



Phases of W-Model:

- Each phase is verified/validated. Dotted line shows that every phase in green is validated/tested through every phase in sky blue.
- Point 1 refers to – Build Test Plan & Test Strategy.
- Point 2 refers to – Scenario Identification.
- Point 3 refers to –Test case preparation from Specification document and design documents.
- Point 4 refers to – Test case preparation from Specification document and design documents.
- Point 5 refers to – review of test cases and update as per the review comments.
- Point 6 refers to – Various testing methodologies such as Unit/integration testing, path testing, equivalence partition, boundary value, specification based testing, security testing, usability testing, performance testing.
- After this, there are regression test cycles and then User acceptance testing.

W - Model



Phases of W-Model

Using W-model helps in ensuring that each phase of the product development is verified and validated. W-model can be divided into a number of stages that includes:

- ① Building **test plan and test strategy** to ensure that the product delivered is tested rigorously before delivery.
- ① Identifying the scenario for the product.
- ① Preparing the test cases using specification and design documents.
- ① Reviewing the test cases and sharing an update on the basis of review comments.
- ① The product is then sent for testing using various testing methodologies such as unit testing, integration testing and specification-based testing, etc.
- ① Once the product is tested rigorously, it, then, undergoes regression test cycles and user acceptance testing.

Testing Techniques Used in W-Model:

1. Regression Testing

2. Static Testing:

Static Testing is further divided into two parts:

(a) Review

(b) Static Analysis

3. Dynamic Testing

REGRESSION TESTING

- **REGRESSION TESTING** is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.
- Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

Static Testing

- **Static Testing** is a type of a Software Testing method which is performed to check the defects in software without actually executing the code of the software application.
- Static testing is performed in early stage of development to avoid errors as it is easier to find sources of failures and it can be fixed easily.
- Static Testing is further divided into two parts:
 - (a) Review
 - (b) Static Analysis

Dynamic Testing

- Dynamic Testing is a kind of software testing technique using which the dynamic behaviour of the code is analysed.
- For Performing dynamic, testing the software should be compiled and executed and parameters such as memory usage, CPU usage, response time and overall performance of the software are analyzed.
- Dynamic testing involves testing the software for the input values and output values are analyzed.
- Dynamic testing is the Validation part of Verification and Validation.

Advantages of W-Model:

- In W-Model there is no strict division between constructive tasks on the left-hand side and the more destructive tasks on the right-hand side.
- During the test phase, the developer is responsible for the removal of defects and the correction of the implementation.
- Emphasis the fact that testing is more than just construction, execution and evaluation of test cases.
- The importance of the tests and the ordering of the individual activities for testing are clear.

Advantages of the W model

- ① Testing can run in parallel with development process
- ① No division between constructive and destructive tasks
- ① Often Developer is responsible for removing defects

Disadvantages Of W-Model:

- The real facts are simplified in this model.
- There is a need for a simple model if all people involved in a project are to accept it.
- For highly critical applications the test activities certainly have higher weighting or at least equal weighting with other activities.

- ⦿ Complex to implement
- ⦿ Resource allocation might not be sufficient in most of the cases
- ⦿ Testing have equal weightage as many activities in the development process

When to use the W model?

- When there are much more activities to do
- Performed when the V model is not enough
- Can be implemented Technical design, architecture and functionality comes to the picture

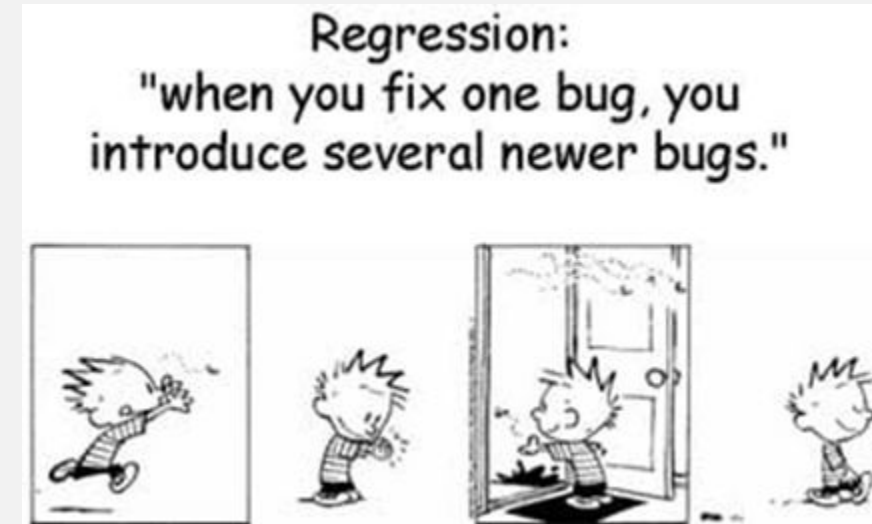
Why to use W model ?

- While it is true that the V model is an effective way to test and reveal results for dynamic test cycles, W models are more effective and help one get a broader view of testing.
- The connection that exists between various stages of testing is much clear with W Model.
- One must choose wisely as the game is not about choosing a v model and w model but delivering an optimum quality product.

Regression Testing

- It involves testing done to make sure none of the changes made over the course of the development process have caused new bugs.
- It also makes sure no old bugs appear from the addition of new software modules over time.

Need of Regression Testing

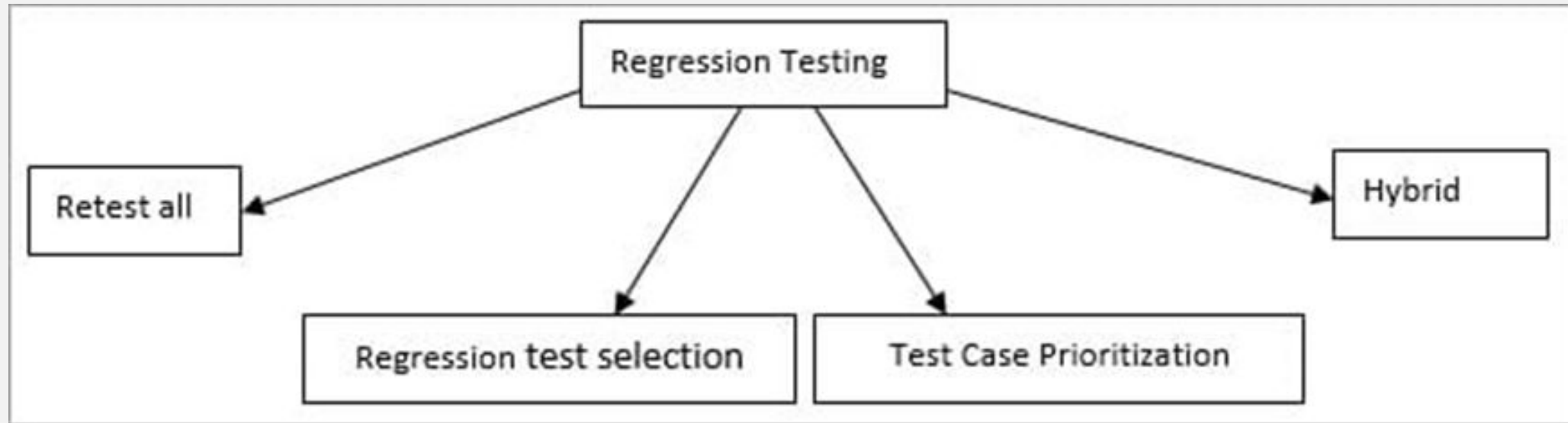


- The **Need of Regression Testing** mainly arises whenever there is requirement to change the code and we need to test whether the modified code affects the other part of software application or not.
- Moreover, regression testing is needed, when a new feature is added to the software application and for defect fixing as well as performance issue fixing.

How to do Regression Testing

- In order **to do Regression Testing** process, we need to first debug the code to identify the bugs. Once the bugs are identified, required changes are made to fix it, then the regression testing is done by selecting relevant test cases from the test suite that covers both modified and affected parts of the code.
- Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary.

Regression Testing can be carried out using the following techniques:

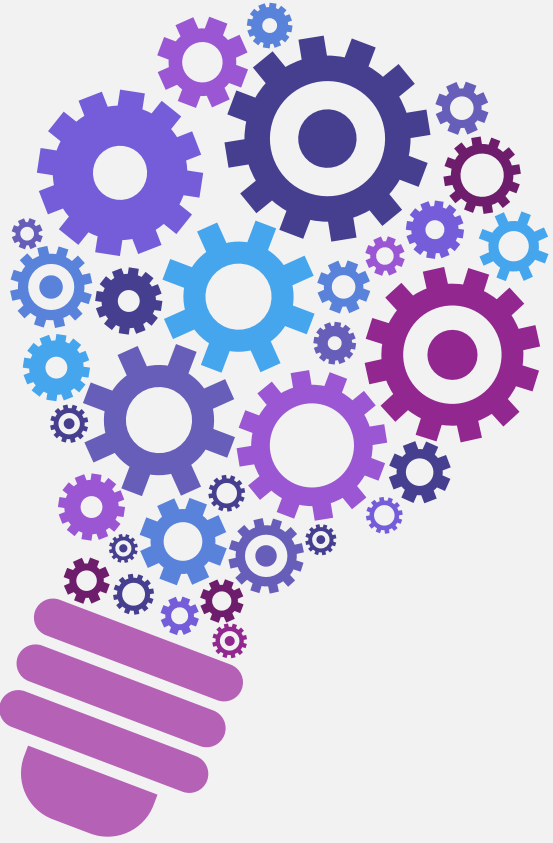


Re-Testing [confirmation testing]

1. Re-testing is the testing for a specific bug after it has been fixed.
2. Testing that runs test cases that failed the last time they were run, in order to verify the success of corrective actions
3. Re-testing can be one which is done for a bug which was raised by QA but could not be found or confirmed by Development and has been rejected. So QA does a re-test to make sure the bug still exists and again assigns it back to them.
4. when entire project is tested & client have some doubts about the quality of testing, Re-Testing can be called. It can also be testing the same application again for better Quality.

Difference between R R

- 1-**Retesting** is done to make sure that bug is fixed and failed functionality is working fine or not, This is kind of verification method followed in testing field for the fixed bugs. Whereas, **Regression** is re-execution of the test cases for unchanged part to see that unchanged functionality is working fine are not.
- 2- **Retesting** is a planned testing while **Regression** is know as the generic [common] testing.
- 3- **Retesting** is only done for failed Test cases while **Regression** is done for passed test cases.
- 4- We should always keep this in mind, **Re-testing has higher priority** than the **regression testing**. But in bigger projects **Retesting and Regression** is done in parallel effort. But never forget importance of both in the success of the project.
-

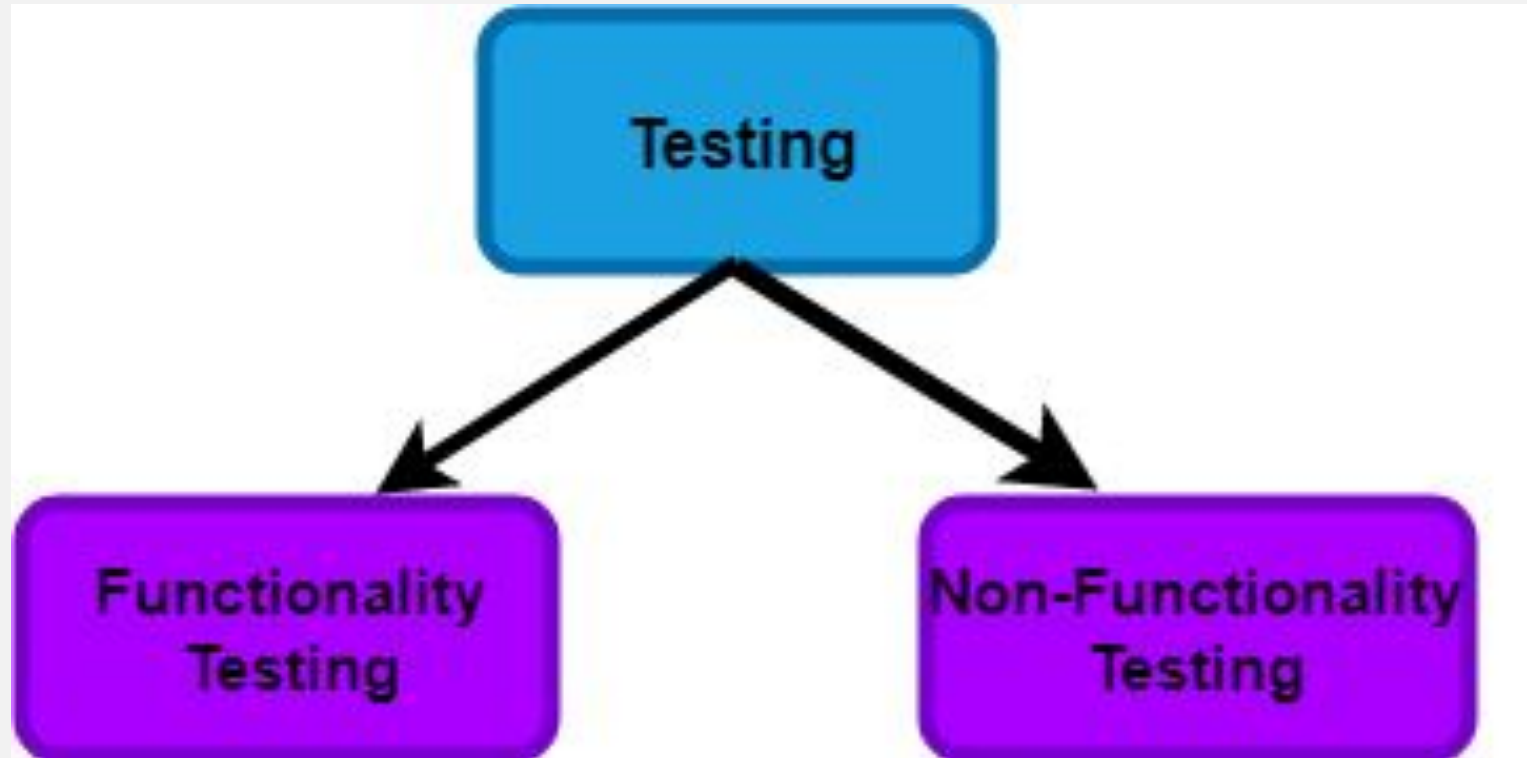


Functional Testing

Testing is to compare the actual result with the expected result.



There are Two type of Testing



What is Functional Testing

FUNCTIONAL TESTING is a type of software testing that validates the software system against the **functional requirements/specifications**. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.

Functional testing verifies **that the software performs its stated functions in a way that the users expect**. The process of functional testing involves a series of tests: Smoke, Sanity, Integration, Regression, Interface, System and finally User Acceptance Testing.

Functional Testing

- It is also known as functional completeness testing.
- Functional Testing involves trying to think of any possible missing functions.
- Testers might make a list of additional functionalities that a product could have to improve it during functional testing.

What do you test in Functional Testing

The prime objective of Functional testing is checking the functionalities of the software system. It mainly concentrates on –

- **Mainline functions** – A test of an application's primary functionalities.
- **Fundamental Usability** – It entails testing the system's basic functionality. It determines whether or not a consumer can easily browse through the screens without difficulty.
- **Accessibility** – Examines the system's usability for the operator.
- **Error Conditions** – The use of testing procedures to detect errors. It determines whether appropriate error messages are presented.

How to do Functional Testing

Identify test input (test data)

Compute the expected outcomes with the selected test input values

Execute test cases

Comparison of actual and computed expected result

Hardware/Software Testing

Hardware Testing Process – How to test products during production

A typical hardware testing process

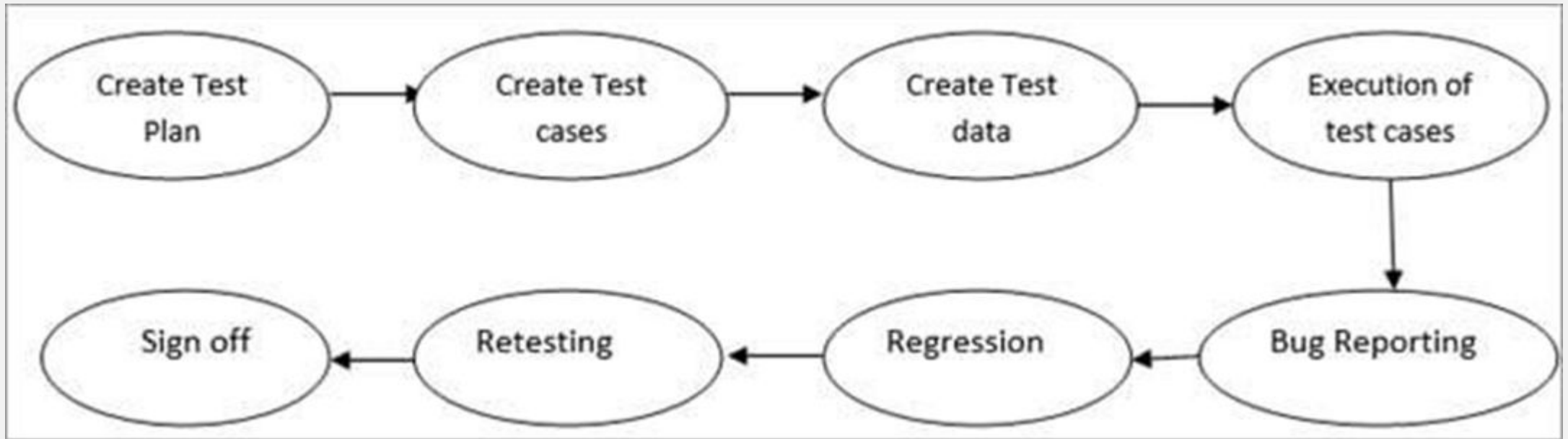
Before testing a piece of hardware, the test engineer should be clear on the purpose of the testing. For example, the specific testing details for a reliability test will be different than the details for a production test. This article discusses the many types of testing that can be applied to a piece of hardware.

1. Create a test plan that lists the test cases used to check functionality against requirements
2. Create a testing environment (e.g., measurement hardware, test software, cabling, fixtures, etc.)
3. Place part into the condition needed for the measurement (apply pressure, voltage, temperature, etc.)
4. Take some measurements
5. Put those measurements through one or more pass/fail criteria
6. Record the results as either summary data or verbose raw plus summary data
7. Repeat 2-5 as needed to sweep through input conditions
8. Create a final report document
9. Declare the part as good or bad
10. Repeat 2-8 for as many parts as need to be tested

Regardless of the purpose of the testing, each test passes through these series of general steps. Of course, the details of the steps will change depending on the type of test.



Steps for performing System testing:

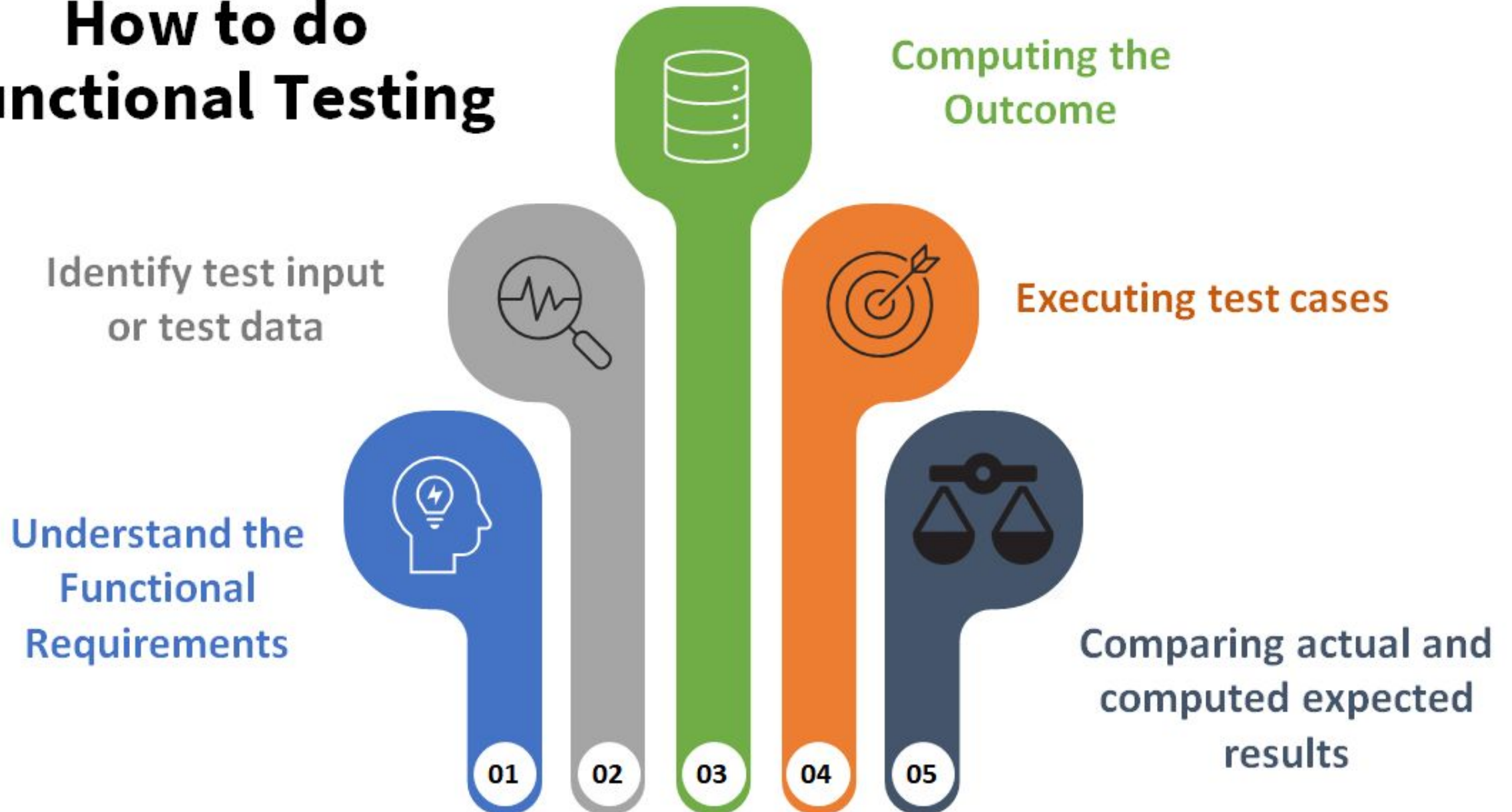


What Types of System Testing Should Testers Use?

There are over 50 different types of system testing. The specific types used by a tester depend on several variables. Those variables include:

- **Who the tester works for** – This is a major factor in determining the types of system testing a tester will use. Methods used by large companies are different than that used by medium and small companies.
- **Time available for testing** – Ultimately, all 50 testing types could be used. Time is often what limits us to using only the types that are most relevant for the software project.
- **Resources available to the tester** – Of course some testers will not have the necessary resources to conduct a testing type. For example, if you are a tester working for a large software development firm, you are likely to have expensive automated testing software not available to others.
- **Software Tester's Education-** There is a certain learning curve for each type of software testing available. To use some of the software involved, a tester has to learn how to use it.
- **Testing Budget** – Money becomes a factor not just for smaller companies and individual software developers but large companies as well.

How to do Functional Testing

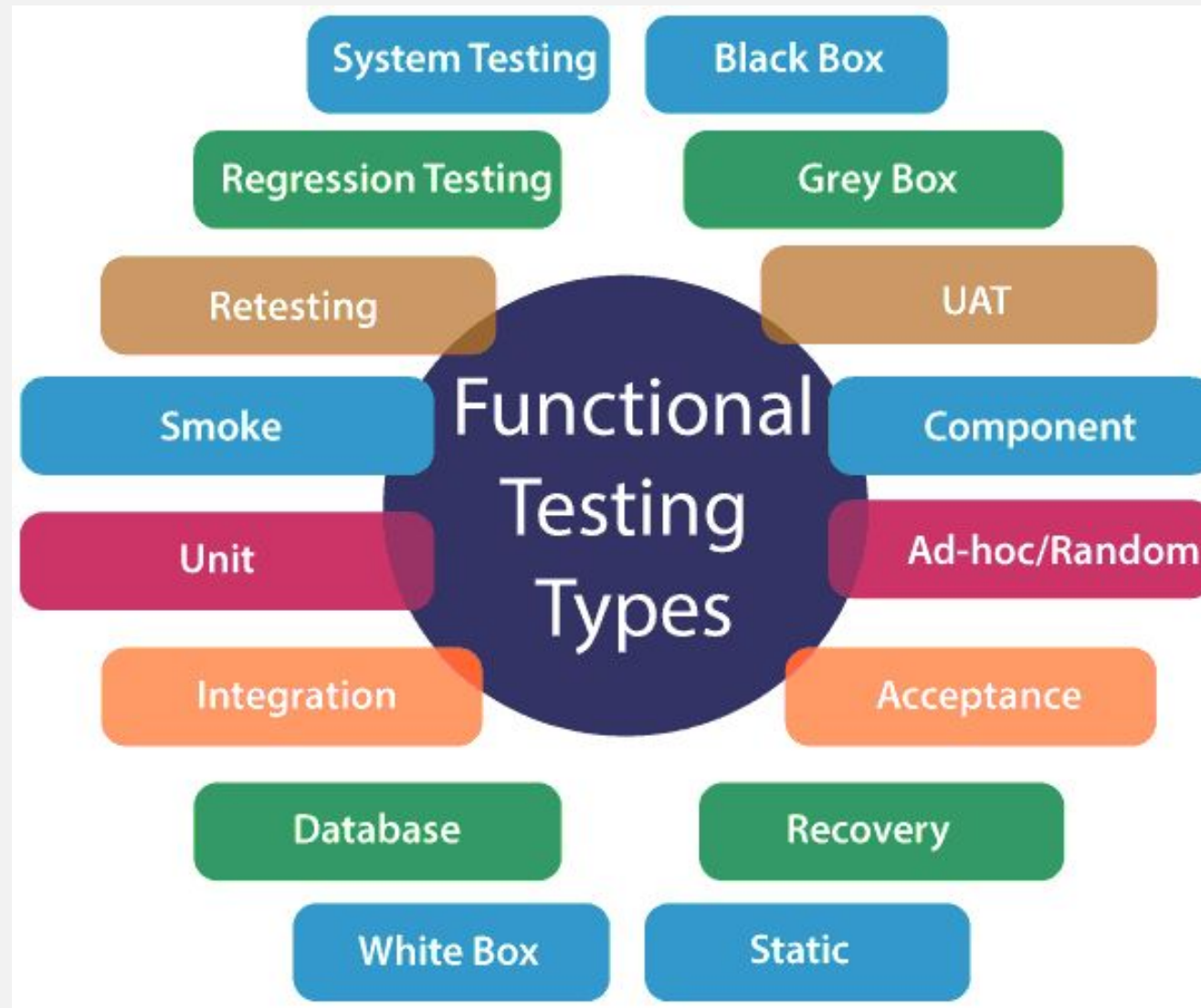


Step performed by Testers

Functional Testing



Types of Functional Testing.



Functional Testing Tools

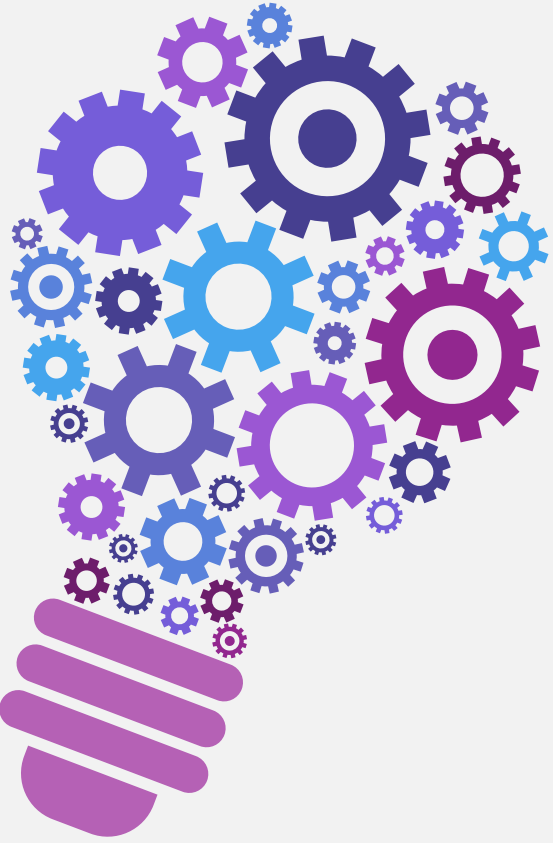
- **Selenium** - Popular Open Source Functional Testing Tool.
- **QTP** - Very user-friendly Functional Test tool by HP.
- **JUnit** - Used mainly for Java Applications and this can be used in Unit and System Testing.
- **soapUI** - This is an open source functional testing tool, mainly used for Web service testing. It supports multiple protocols such as HTTP, SOAP, and JDBC.
- **Watir** - This is a functional testing tool for web applications. It supports tests executed at the web browser and uses a ruby scripting language

Advantages of Functional Testing

- It produces a defect-free product.
- It ensures that the customer is satisfied.
- It ensures that all requirements met.
- It ensures the proper working of all the functionality of an application/software/product.
- It ensures that the software/ product works as expected.
- It ensures security and safety.
- It improves the quality of the product.

Disadvantages of functional testing

- Functional testing can miss a critical and logical error in the system.
- This testing is not a guarantee of the software to go live.
- The possibility of conducting redundant testing is high in functional testing.



Non -Functional Testing

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



What is Non Functional Testing ?

- Non-Functional testing can be defined as a type of software testing to check non-functional aspects (performance, usability, reliability, etc.) of a software application.
- It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.
- An excellent example of non-functional test would be to check how many people can simultaneously login into a software.
- Non-functional testing is equally important as functional testing and affects client satisfaction.

Objectives of Non Functional Testing

- Non-functional testing **should increase** usability, efficiency, maintainability, and portability of the product.
- Helps to **reduce** production risk and cost associated with non-functional aspects of the product.
- **Optimize** the way product is installed, setup, executes, managed and monitored.
- Collect and produce **measurements**, and metrics for internal research and development.
- Improve **and enhance knowledge** of the product behavior and technologies in use.

Characteristics of Non Functional Testing

- Non-functional testing should be **measurable**, so there is no place for subjective characterization like good, better, best, etc.
- **Exact numbers** are unlikely to be known at the start of the requirement process
- Important to **prioritize** the requirements
- Ensure that **quality** attributes are identified correctly in Software Engineering.

Functional Vs Non Functional Testing

Functional Texting	Non - Functional Testing
Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system.
It is based on customer's requirements.	It focusses on customer's expectation.
Functional testing is executed first	Non-functional testing should be performed after functional testing
Carried out to validate software actions.	It is done to validate the performance of the software.
Business requirements are the inputs to functional testing	Performance parameters like speed, scalability are inputs to non-functional testing.
Functional testing describes what the product does	Nonfunctional testing describes how good the product works
Eg : Check login functionality.	Eg : The dashboard should load in 2 seconds.
Examples of Functional testing are <ul style="list-style-type: none">• Unit Testing• Integration Testing• White Box Testing• Black Box Testing	Examples of Non - Functional testing are <ul style="list-style-type: none">• Performance Testing• Load testing• Security Testing• Compatibility Testing

Non Functional Testing Parameters

1. Security

2. Availability

3. Efficiency

4. Integrity

5. Reliability

6. Survivability

7. Usability

8. Flexibility

9. Scalability

10. Reusability

11. Interoperability

12. Portability

1. Security

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

2. Reliability

The extent to which any software system continuously performs the specified functions without failure. This is tested by **Reliability Testing**.

3. Survivability

The parameter checks that the software system continues to function and recovers itself in case of system failure. This is checked by **Recovery Testing**.

4. Availability

The parameter determines the degree to which user can depend on the system during its operation. This is checked by **Stability Testing**

5. Usability

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system. This is checked by **Usability Testing**

6. Scalability

The term refers to the degree in which any software application can expand its processing capacity to meet an increase in demand. This is tested by **Scalability Testing**

7. Interoperability

This non-functional parameter checks a software system interfaces with other software systems. This is checked by **Interoperability Testing**.

8. Efficiency

The extent to which any software system can handles capacity, quantity and response time.

9. Flexibility

The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

10. Portability

The flexibility of software to transfer from its current hardware or software environment.

11. Reusability

It refers to a portion of the software system that can be converted for use in another application.

12. Integrity

It measures the how high the source code's quality is when it is passed on to the QA, and is affected by how extensively the code was unit tested and integration tested.

Following are the most common Types of Non Functional Testing :

1. Performance Testing
2. Load Testing
3. Compatibility Testing
4. Stress Testing
5. Scalability Testing
6. Reliability Testing
7. Portability Testing
8. Efficiency Testing
9. Security Testing
10. Installation Testing

Non Functional Testing Examples

answerit

Test Case #	Test Cases	Domain
T01	The system should be able to handle thousands of request simultaneously without any server slow down.	
T02	The system authorizes and validates all the payments from the passengers through a secured payment gateway.	
T03	The system can be scaled up to handle lakhs of request together and multiple administrator request simultaneously without any deadlock.	
T04	All web images should have alt tags	
T05	This application should work in all browsers like chrome, firefox, safari, edge,etc.	

Non Functional Testing Examples

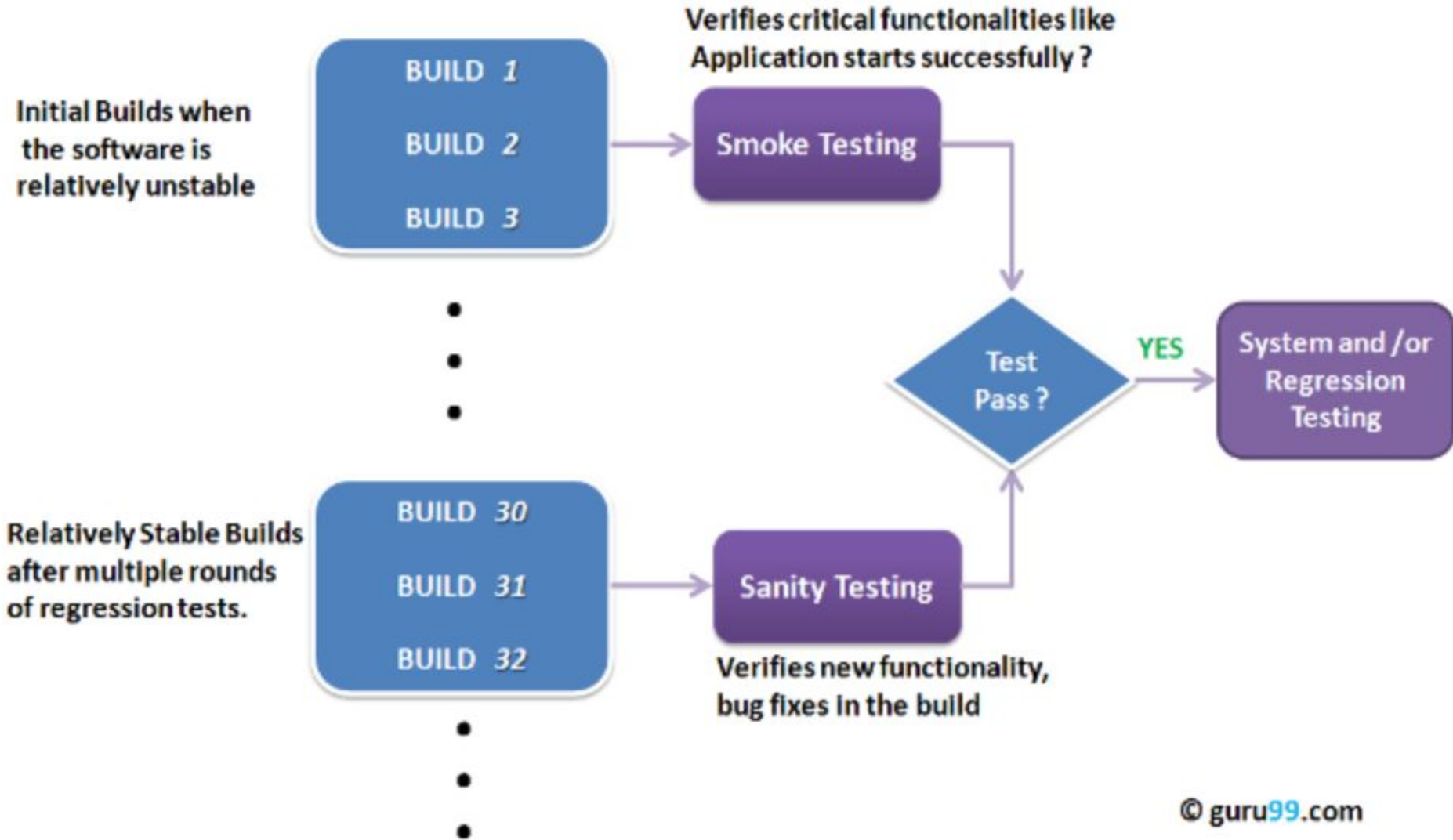
Test Case #	Test Cases	Domain
T06	Application load time should not be more than 5 sec up to 1000 users accessing it simultaneously	
T07	The Software should be installable on all versions of Windows and Mac	
T08	Users must change their initial login authentication information after first successful signin.	
T09	There should be role based authentication mechanism	

Non Functional Testing Examples

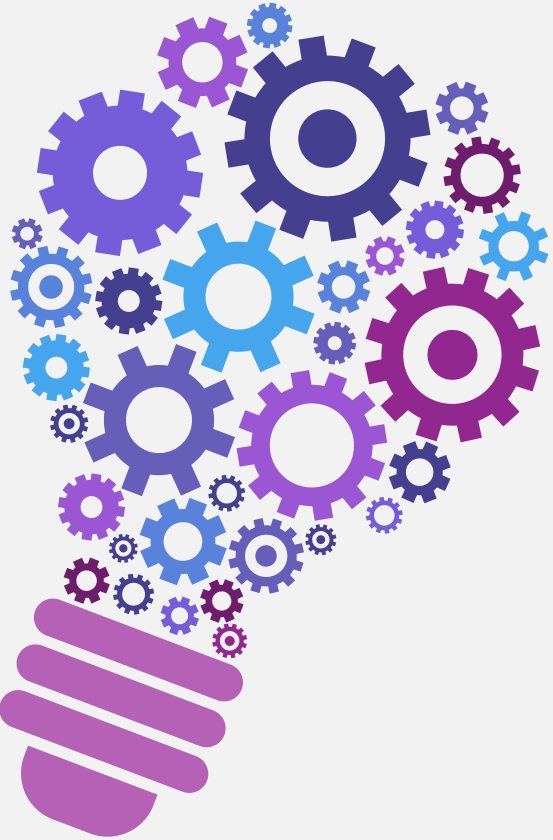
Test Case #	Test Cases	Domain
T01	The system should be able to handle thousands of request simultaneously without any server slow down.	Performance
T02	The system authorizes and validates all the payments from the passengers through a secured payment gateway.	Reliability
T03	The system can be scaled up to handle lakhs of request together and multiple administrator request simultaneously without any deadlock.	Scalability
T04	All web images should have alt tags	Accessibility
T05	This application should work in all browsers like chrome, firefox, safari, edge,etc.	Compatibility

Non Functional Testing Examples

Test Case #	Test Cases	Domain
T06	Application load time should not be more than 5 sec up to 1000 users accessing it simultaneously	Performance
T07	The Software should be installable on all versions of Windows and Mac	Compatibility
T08	Users must change their initial login authentication information after first successful signin.	Security
T09	There should be role based authentication mechanism	Security



Smoke Testing vs Sanity Testing



1. Component Testing

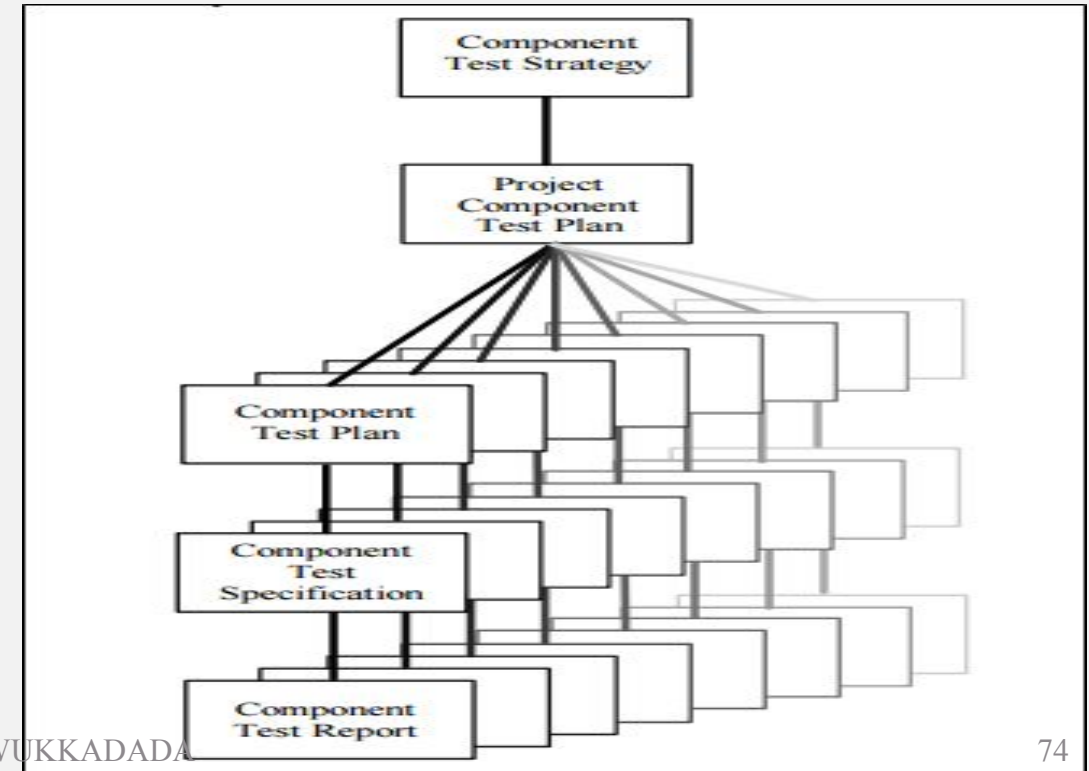
(Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components.)



What is Component Testing?

- Component testing is defined as a software testing type, in which the testing is performed on each individual component separately without integrating with other components.
- It's also referred to as Module Testing when it is viewed from an architecture perspective. Component Testing is also referred to as Program Testing or Module Testing.
- Generally, any software as a whole is made of several components. Component Level Testing deals with testing these components individually.
- It's one of most frequent black box testing types which is performed by QA Team.
- When component testing is done **without isolation** with other components of the software then it is called as **component testing in large**. This happens when there is a dependency on the functionality flow of the components and thus we can't isolate them.
- If the components on which we have dependency are not developed yet, then **we use dummy** objects in place of the actual components. These dummy objects are **the stub (called function) and driver (calling function)**.

- Component testing may be done in isolation from rest of the system depending on the [development life cycle](#) model chosen for that particular application.
- In such case the missing software is replaced by **Stubs** and **Drivers** and simulate the interface between the software components in a simple manner.
- As per the below diagram, there will be a test strategy and test plan for component testing. Where each and every part of the software or application is considered individually.
- For each of this component a [Test Scenario](#) will be defined, which will be further brought down into a High Level Test Cases -> Low Level detailed Test Cases with Prerequisites.



□ The most common reason for different perception of **Component** testing are-

- Type of Development Life Cycle Model Chosen
 - Complexity of the software or application under test
 - Testing with or without isolation from rest of other component in software or application.
-
- As we know Software Test Life Cycle Architecture has lots many test-artifacts (Documents made, used during testing activities).
 - Among many tests – artifacts, it's the Test Policy & Test Strategy which defines the types of testing, depth of testing to be performed in a given project.
 - Component testing plays a very important role in finding the bugs. Before we start with the integration testing it's always preferable to do the component testing in order to ensure that each component of an application is working effectively.

Objective of Component Testing:

- To verify the input and output behavior of the system.
- To check the usability of each component.
- To test the user comprehensibility of the software.
- To test the state of the each components of the system.

How to write component test cases?

- The test cases for component testing are derived from work products, for instance, software design or the data model.
- Each component is tested through a sequence of test cases where each test case covers a specific combination of input/output i.e. partial functionality.

Requirement Analysis:

User requirement related to each component is observed.

Test Planning:

Test is planned according to the analysis of the requirements of the user.

Test Specification:

In this section it is specified that which test case must be run and which test case should be written.

Test Execution:

Once the test cases are specified according to the user requirements, test cases are executed.

Test Recording:

Test recording is the having record of the defects that are detected.

Test Verification:

Test verification is the process to determine whether the product meet specification.

Completion:

This is the last phase of the testing process in which the result is analyzed.

Component Testing Test Strategy

Depending upon the depth of testing level, component testing is divided into two parts:

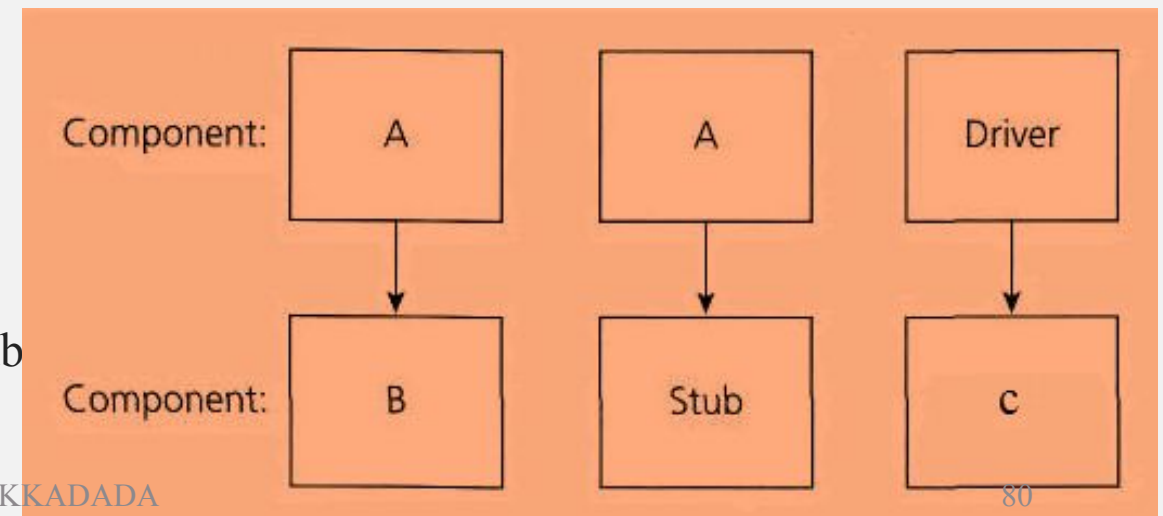
1. Component Testing in Small (CTIS)
2. Component Testing in Large (CTIL)

1. CTIS – Component Testing in Small

Component testing may be done with or without isolation of rest of other components in the software or application under test. If it's performed with the isolation of other component, then it's referred as Component Testing in Small.

2. CTIS – Component Testing in Small

- Component testing done without isolation of other components in the software or application under test is referred as **Component Testing Large**.
- Let's take an example to understand it in a better way. Suppose there is an application consisting of three components say **Component A**, **Component B**, and **Component C**.
- The developer has developed the component B and wants it tested. But in order to **completely** test the component B, few of its functionalities are dependent on component A and few on component C.
- Functionality Flow: **A -> B -> C** which means there is a dependency to B from both A & C, as per the diagram stub is the **called function**, and the driver is the **calling function**.
- But the component A and component C has not been developed yet. In that case, to test the component B completely, we can replace the component A and component C by stub and drivers as required. So basically, component A & C are replaced by stub & driver's which acts as a dummy object till they are actually developed.





DRIVER-
CALLING FUNCTION



STUB-
CALLED FUNCTION

Drivers and Stubs for Unit Testing

- Driver
 - A simple **main program** that accepts test case data, passes such data to the component being tested, and prints the returned results
- stub
 - Serve to replace modules that are subordinate to **(called by)** the component to be tested
 - It uses the module's exact interface, may do minimal data manipulation, provides verification of entry, and returns control to the module undergoing testing
- Drivers and stubs both represent overhead
 - Both must be written but don't constitute part of the installed software product

Drivers and stubs

[READ]

- It is always a good idea to develop and test software in "pieces". But, it may seem impossible because it is hard to imagine how you can test one "piece" if the other "pieces" that it uses have not yet been developed (and vice versa). To solve this kind of difficult problems we use stubs and drivers. In white-box testing, we must run the code with predetermined input and check to make sure that the code produces predetermined outputs. **Often Testers write stubs and drivers for white-box testing.**

Driver for Testing:

Driver is a the piece of code that passes test cases to another piece of code. Test Harness or a test driver is supporting code and data used to provide an environment for testing part of a system in isolation. **It can be called as as a software module** which is used to invoke a module under test and provide test inputs, control and, monitor execution, and report test results or most simplistically a line of code that calls a method and passes that method a value.

For example, if you wanted to move a fighter on the game, the **driver code** would be `moveFighter (Fighter, LocationX, LocationY);`

This driver code would likely be called from the main method. A white-box test case would execute this driver line of code and check "`fighter.getPosition()`" to make sure the player is now on the expected cell on the board.

● Stubs for Testing:

A Stub is a dummy procedure, module or unit that stands in for an unfinished portion of a system.

Four basic types of Stubs for Top-Down Testing are:

- 1 Display a trace message
- 2 Display parameter value(s)
- 3 Return a value from a table
- 4 Return table value selected by parameter

A stub is a computer program which is used as a substitute for the body of a software module that is or will be defined elsewhere or a dummy component or object used to simulate the behavior of a real component until that component has been developed.

Ultimately, the dummy method would be completed with the proper program logic. However, developing the stub allows the programmer to call a method in the code being developed, even if the method does not yet have the desired behavior.

Stubs and drivers are often viewed as throwaway code. However, they do not have to be thrown away: Stubs can be "filled in" to form the actual method. Drivers can become automated test cases.

- The concept of Stubs and Drivers are mostly used in the case of component testing. Component testing may be done in isolation with the rest of the system depending upon the context of the development cycle.

Stubs and drivers are used to replace the missing software and simulate the interface between the software components in a simple manner.

Suppose you have a function (Function A) that calculates the total marks obtained by a student in a particular academic year. Suppose this function derives its values from another function (Function b) which calculates the marks obtained in a particular subject.

You have finished working on Function A and wants to test it. But the problem you face here is that you can't seem to run the Function A without input from Function B; Function B is still under development. In this case, you create a dummy function to act in place of Function B to test your function. This dummy function gets called by another function. Such a dummy is called a Stub.

To understand what a driver is, suppose you have finished Function B and is waiting for Function A to be developed. In this case you create a dummy to call the Function B. This dummy is called the driver.

Advantages and Disadvantages

Advantages

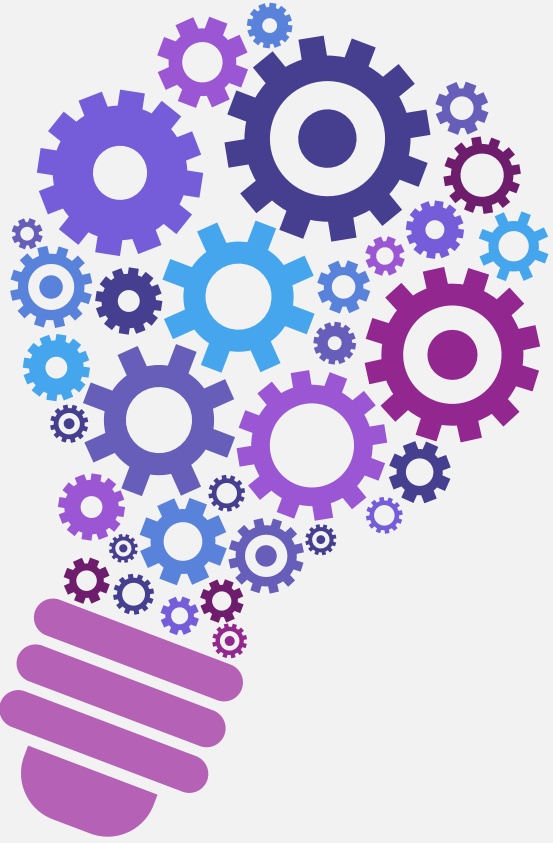
1. It finds the defects in the module and verifies the functioning of the software.
2. It helps in faster delivery.
3. More reliable systems because previously tested components are used.
4. It leads to re-usability of software which provides a lot of benefits.
5. Reduces the development cycle time.
6. Helps in reducing the project cost.
7. Leads to a significant increase in productivity.

Disadvantages

1. Less control over the evolution of the system.
2. There is a need to compromise the requirements.

Summary:

- In Software Engineering, Component testing plays a crucial role in finding the bugs. Before we start with the [Integration Testing](#), it's always recommended to perform the component testing in order to ensure that each component of an application is working effectively.
- Integration Testing is followed by the component testing. Component testing also referred as module testing in some references.



Integration Testing

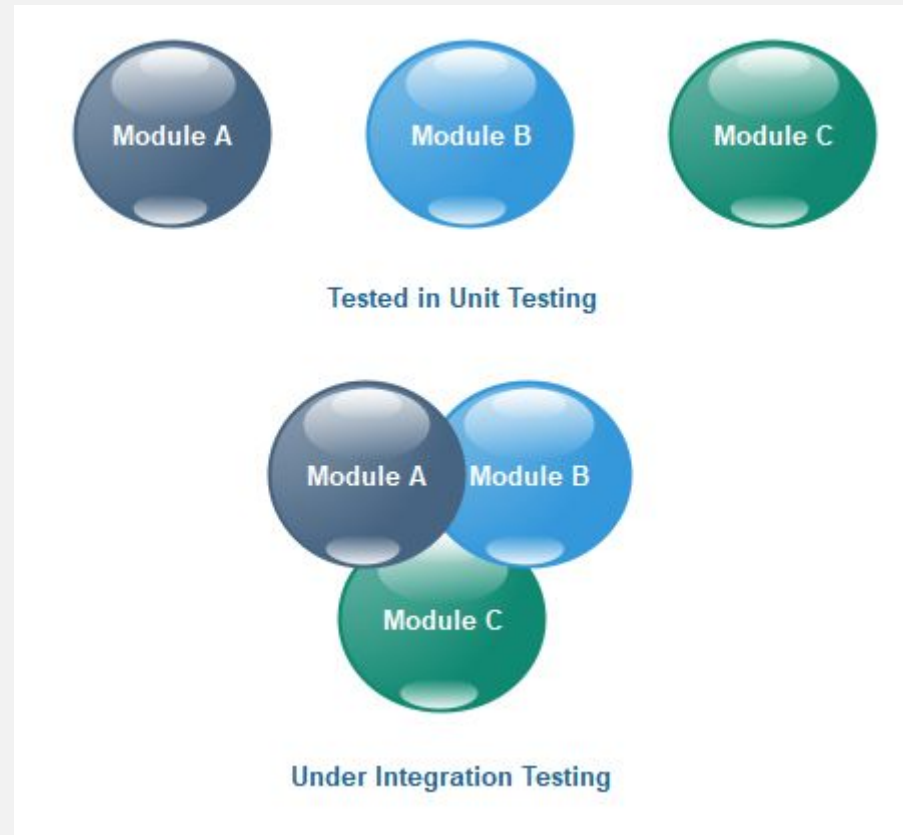
(Integration Testing is a level of software testing where individual units are combined and tested to verify if they are working as they intend to when integrated)



What is Integration testing?

- Software modules are integrated logically.
- Tested as a group.
- Purpose is to expose defects between different software modules.

A diagrammatic representation.



Why should we do Integration Testing?

- Unit level testing alone is not enough to ensure reliability.
- Some user requested unit level changes in modules can cause issues.
- Just unit testing can lead to several problems that are overlooked.

Advantages of Integration Testing

- Work as Intended
- Eases load on testers
- Tests bonds between modules
- Promotes reliability between modules, APIs and external applications
- Is more efficient
- Improves speed and efficacy of tests

Types of Integration Testing

- **Incremental Testing**

Top – Down Approach

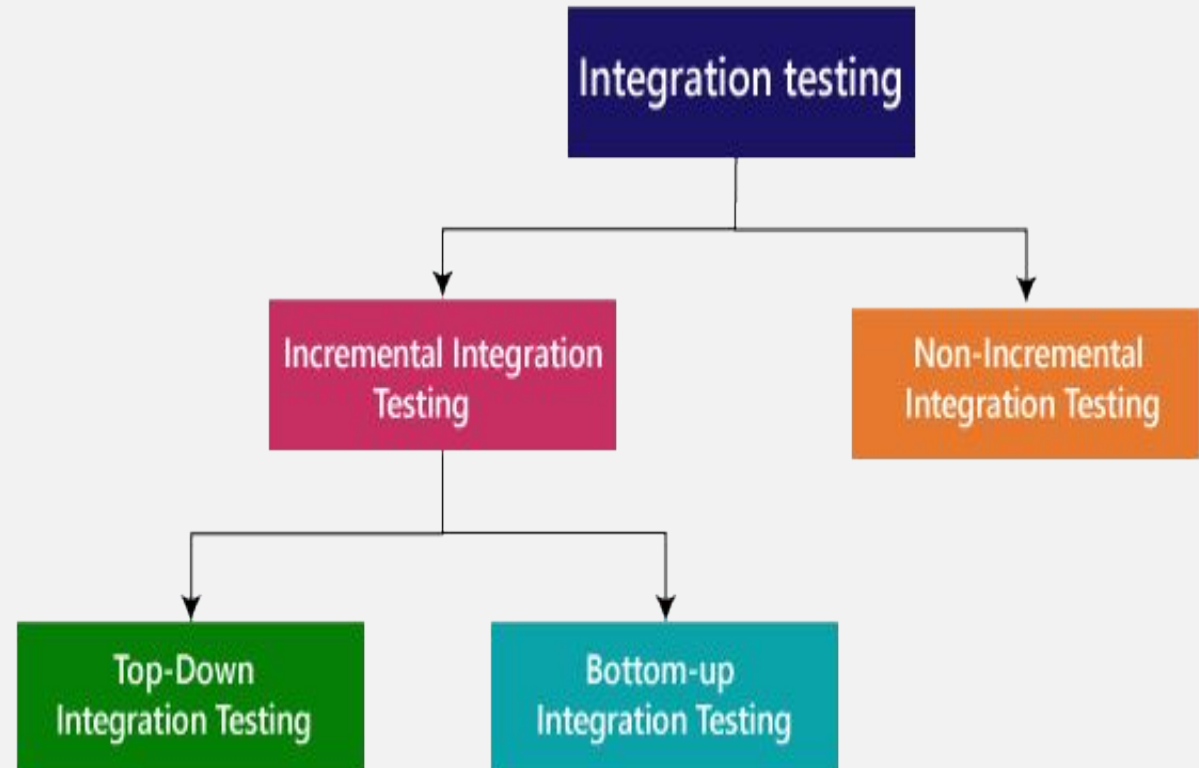
Bottom – Up Approach

Hybrid Method

- **Non-Incremental Testing**

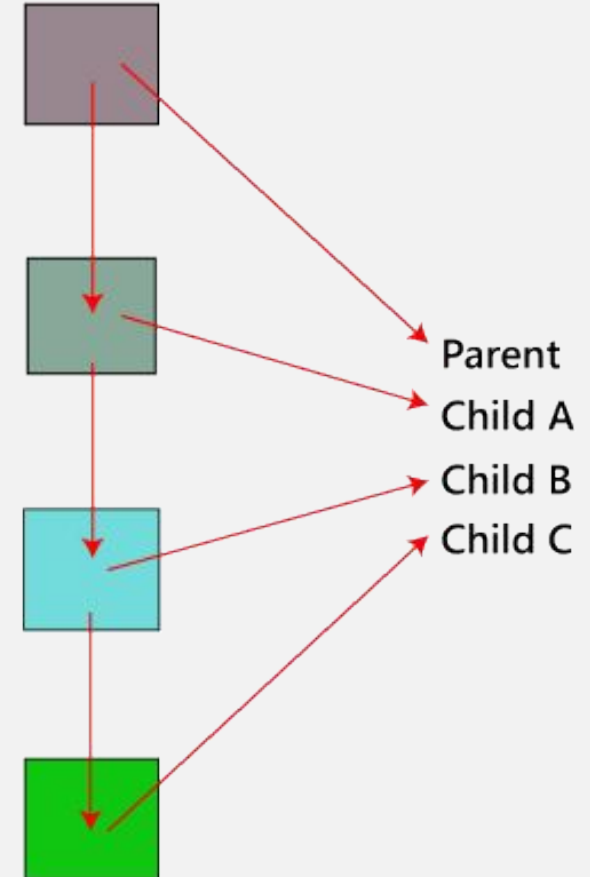
Big-Bang Method

Stub and Driver Method.



Top-down Approach

- Higher level modules are tested first.
- Flaws can be detected earlier and fixed.



Bottom – Up Approach

- Lower level modules are tested first.
- Data is checked from bottom to top
- Fault localization is easier.

Hybrid Testing

- Both top-down and bottom-up approaches are tested simultaneously.
- Less possibility of occurrence of defect.
- Most time reducing method.

Big-bang method

- Used for small sized softwares.
- Testing is done via integration of all modules at once.
- If all of the components in the unit are not completed, the integration process will not execute.

Stub and driver method

Are used as dummy programs.

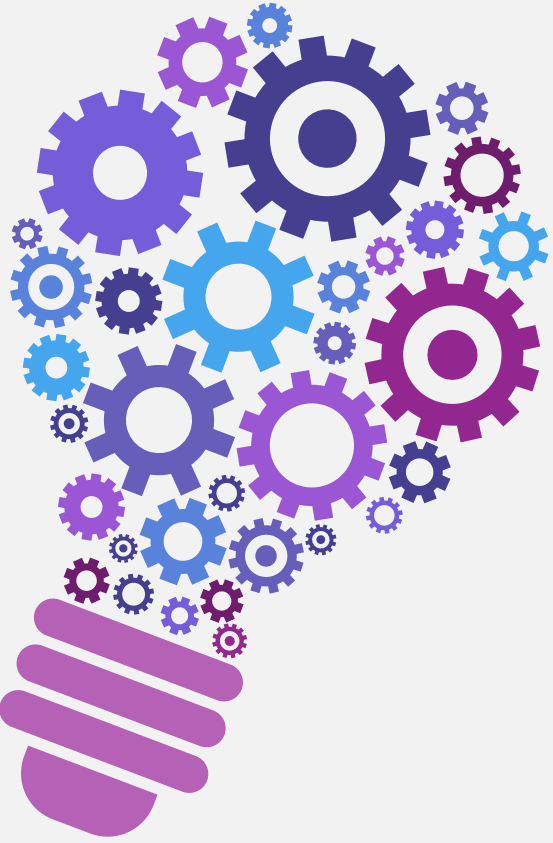
Act as substitutes for the missing models.

How to do Integration testing?

- Prepare and design the test scenarios, cases and scripts.
- Execute the cases and identify the defects.
- Re-testing

Example

- A website development project is assigned to a team of developers.
- Each developer is assigned a module.
- They developed the module on their own machines.
- Unit-testing was done to identify the defects and they were fixed.. and each module was completed.
- Then, integration testing is done ensure the overall project is compatible and working.
- Many bugs were found and incompatibility issues were solved.
- The project was completed.

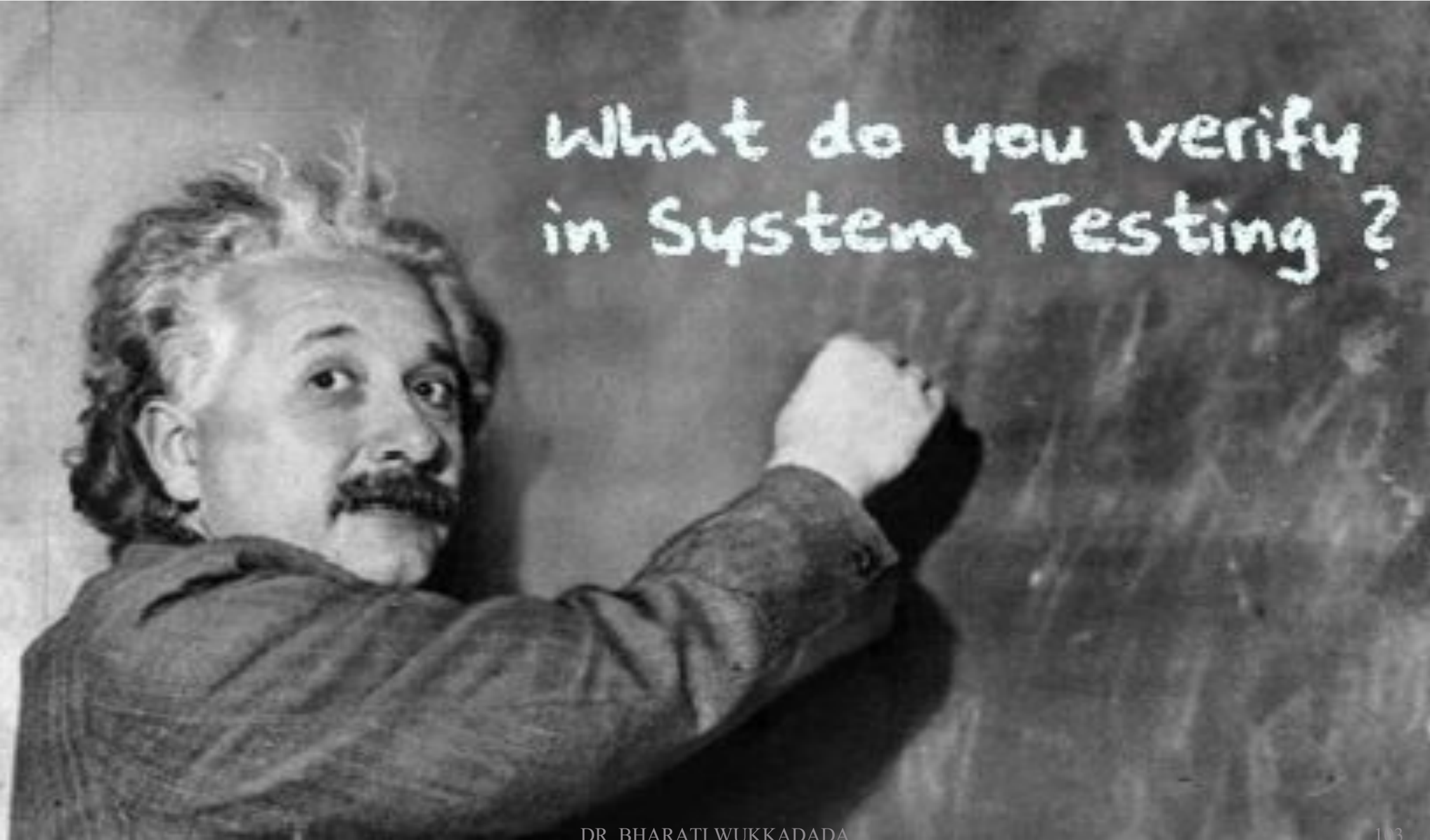


4. System Testing



What is System Testing?

- System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.
- System testing is carried out by specialists testers or independent testers.
- System testing should investigate both functional and non-functional requirements of the testing.

A black and white photograph of Albert Einstein, looking over his shoulder with a questioning expression, pointing his right index finger at a chalkboard. On the chalkboard, the text "What do you verify in System Testing?" is written in a white, chalk-like font. The background is a dark, textured chalkboard surface.

What do you verify
in System Testing ?

What do you verify in System Testing?

System Testing involves testing the software code for following

- Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
- Verify thorough testing of every input in the application to check for desired outputs.
- Testing of the user's experience with the application.

Different Types of System Testing

- There are more than 50 types of System Testing.
 - Below we have listed types of system testing a large software development company would typically use
1. Usability Testing
 2. Load Testing
 3. Regression Testing
 4. Functional Testing
 5. Hardware/Software Testing

Usability Testing

Usability Testing also known as **User Experience(UX)** Testing, is a testing method for measuring how easy and user-friendly a software application is. A small set of target end-users, use software application to expose usability defects. Usability testing mainly focuses on user's ease of using application, flexibility of application to handle controls and ability of application to meet its objectives.

Usability Testing

determines whether an application is

Useful

Findable

Accessible

Usable

Desirable

The goal of this testing is to satisfy users and it mainly concentrates on the following parameters of a system:

- The effectiveness of the system
- Efficiency
- Accuracy
- User Friendliness

How to do Usability Testing: Complete Process



©guru99.com

Load Testing

capacity of an application

Load Testing is a non-functional software testing process in which the performance of software application is tested under a specific expected load. It determines how the software application behaves while being accessed by multiple users simultaneously. The goal of Load Testing is to improve performance bottlenecks and to ensure stability and smooth functioning of software application before deployment.



This testing usually identifies -

- The maximum operating capacity of an application
- Determine whether the current infrastructure is sufficient to run the application
- Sustainability of application with respect to peak user load
- Number of concurrent users that an application can support, and scalability to allow more users to access it.

Goals of Load Testing

Loading testing identifies the following problems before moving the application to market or Production:

- Response time for each transaction
- Performance of System components under various loads
- Performance of Database components under different loads
- Network delay between the client and the server
- Software design issues
- Server configuration issues like a Web server, application server, database server etc.
- Hardware limitation issues like CPU maximization, memory limitations, network bottleneck, etc.

Strategies of Load Testing

There are many numbers of ways to perform load testing. Following are a few load testing strategies :-



- **Manual Load Testing:** This is one of the strategies to execute load testing, but it does not produce repeatable results, cannot provide measurable levels of stress on an application and is an impossible process to coordinate.
- **In house developed load testing tools:** An organization, which realizes the importance of load testing, may build their own tools to execute load tests.
- **Open source load testing tools:** There are several load testing tools available as open source that are free of charge. They may not be as sophisticated as their paid counterparts, but if you are on a budget, they are the best choice.
- **Enterprise-class load testing tools:** They usually come with capture/playback facility. They support a large number of protocols. They can simulate an exceptionally large number of users.



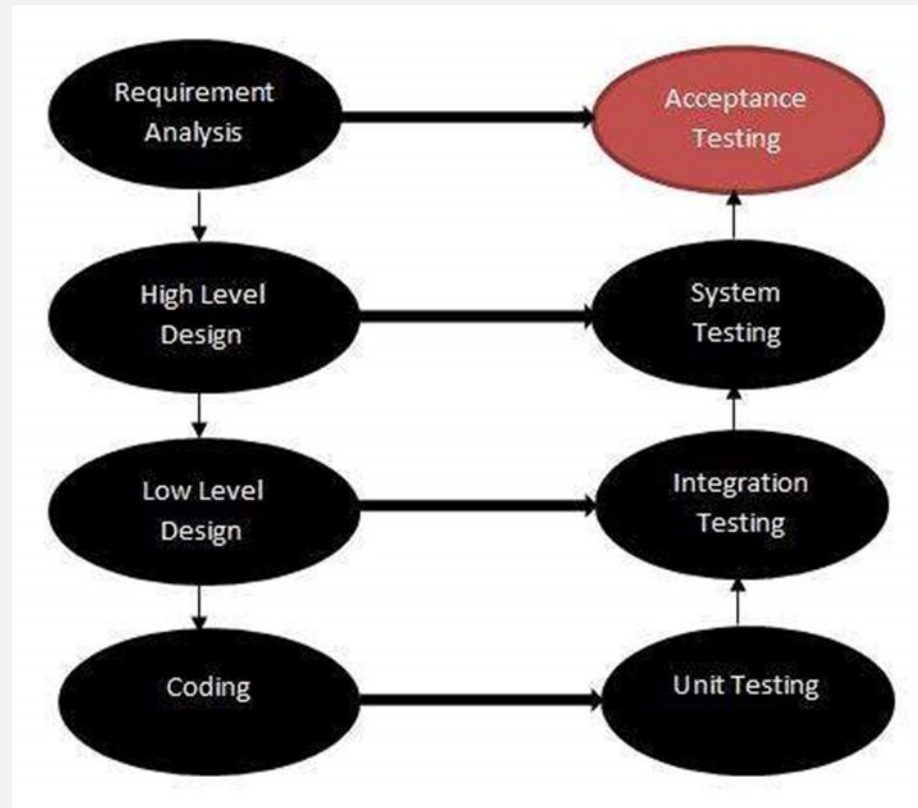
5. Acceptance Testing

(Acceptance testing is formal testing based on user requirements and function processing.)



Acceptance Testing - In SDLC

- The following diagram explains the fitment of acceptance testing in the software development life cycle.



Introduction

- Acceptance testing is formal testing based on user requirements and function processing. It determines whether the software is conforming specified requirements and user requirements or not. It is conducted as a kind of Black Box testing where the number of required users involved testing the acceptance level of the system. It is the fourth and last level of software testing.

Types of Acceptance Testing:

- **1. User Acceptance Testing (UAT):**

User acceptance testing is used to determine whether the product is working for the user correctly. Specific requirements which are quite often used by the customers are primarily picked for the testing purpose. This is also termed as End-User Testing.

- **2. Business Acceptance Testing (BAT):**

BAT is used to determine whether the product meets the business goals and purposes or not. BAT mainly focuses on business profits which are quite challenging due to the changing market conditions and new technologies so that the current implementation may have to be changed which result in extra budgets.

3. **Contract Acceptance Testing (CAT):**

CAT 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.

Here is a **contract 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.

Sometimes, this contract happens before the product goes live. There should be a well defined contract in terms of the period of testing, areas of testing, conditions on issues encountered at later stages, payments, etc.

- **4.Regulations Acceptance Testing (RAT):**

RAT is used to determine whether the product violates the rules and regulations that are defined by the government of the country where it is being released. This may be unintentional but will impact negatively on the business.

- Generally, the product or application that is to be released in the market, has to go under RAT, as different countries or regions have different rules and regulations defined by its governing bodies. If any rules and regulations are violated for any country then that country or the specific region then the product will not be released in that country or region. If the product is released even though there is a violation then only the vendors of the product will be directly responsible.

- **5.Operational Acceptance Testing (OAT):**

OAT is used to determine the operational readiness of the product and is a non-functional testing. It mainly includes testing of recovery, compatibility, maintainability, reliability etc.

OAT assures the stability of the product before it is released to the production.

- **6.Alpha Testing:**

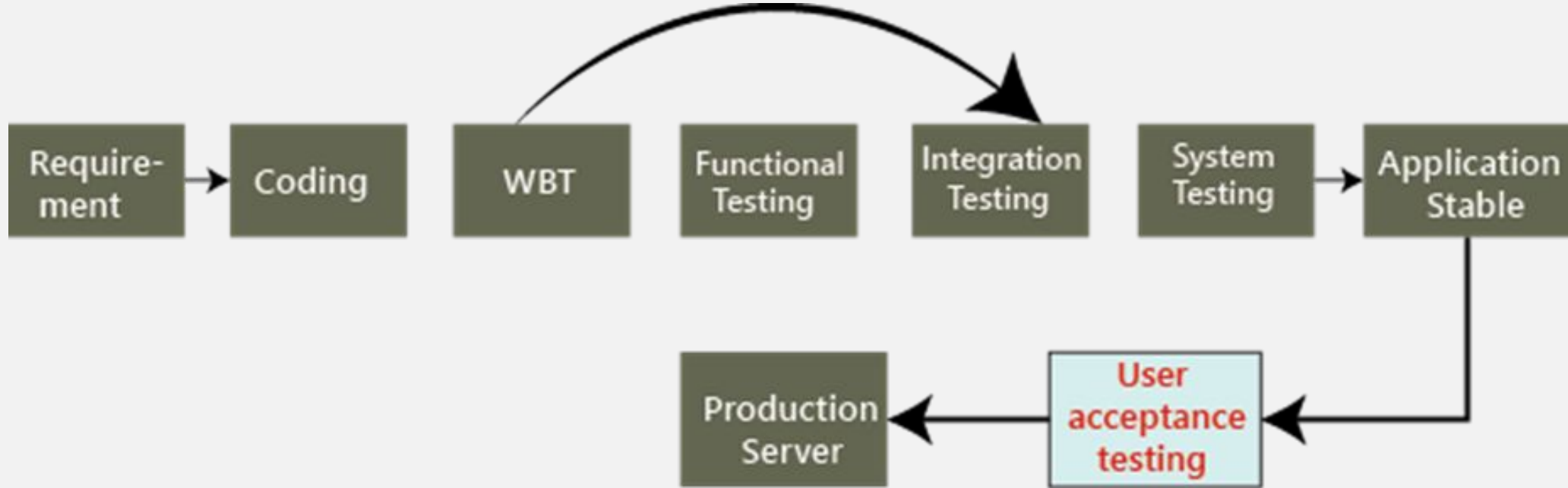
Alpha testing is used to determine the product in the development testing environment by a specialized testers team usually called alpha testers.

- **7.Beta Testing:**

Beta testing is used to assess the product by exposing it to the real end-users, usually called beta testers in their environment. Feedback is collected from the users and the defects are fixed. Also, this helps in enhancing the product to give a rich user experience

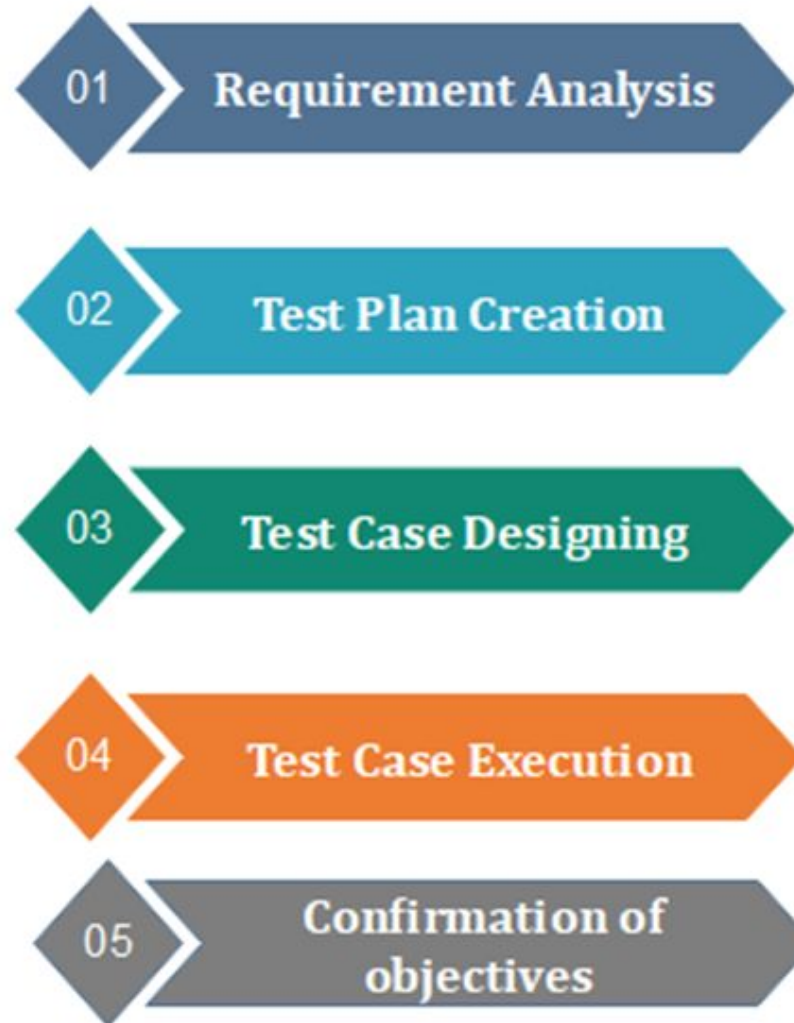
Reason behind Acceptance Testing

- Once the software has undergone through Unit Testing, Integration Testing and System Testing so, Acceptance Testing may seem redundant, but it is required due to the following reasons.
 - During the development of a project if there are changes in requirements and it may not be communicated effectively to the development team.
 - Developers develop functions by examining the requirement document on their own understanding and may not understand the actual requirements of the client.
 - There's maybe some minor errors which can be identified only when the system is used by the end user in the actual scenario so, to find out these minor errors, acceptance testing is essential.



- Once the application is bug-free, we handover it to the customer, no customer accept the application blindly before using it. Hence, they do one round of testing for their satisfaction, which is known as user acceptance testing.

Steps to Perform Acceptance Testing



Tools used in Acceptance Testing

- **Acceptance** Testing can be done by using several tools; some are given below:
- done by using several tools; some are given below:

1. **Watir:**

- Acceptance testing uses this tool for the execution of automated browser-based test cases. It uses Ruby language for the inter-process communication.

2. **Fitness tool:**

- This tool is used to enter input values and generate test cases automatically. The user needs to input values, these values used by the tool to execute test cases and to produce output. It uses Java language for the inter-process communication. This tool makes it easy to create test cases as well as record them in the form of a table.

Advantages of Acceptance Testing

- It increases the satisfaction of clients as they test application itself.
- The quality criteria of the software is defined in an early phase so that the tester has already decided the testing points. It gives a clear view to testing strategy.
- The information gathered through acceptance testing used by stakeholders to better understand the requirements of the targeted audience.
- It improves requirement definition as client tests requirement definition according to his needs.

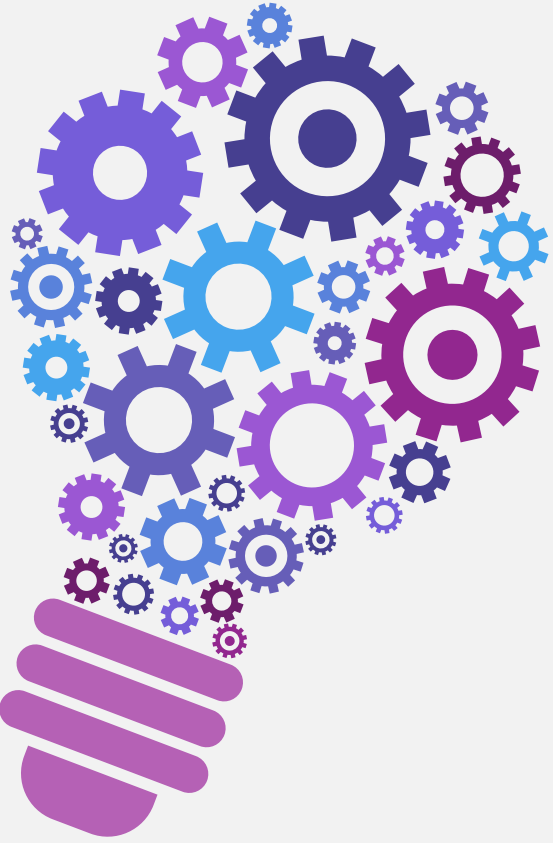
Disadvantages of Acceptance Testing

- According to the testing plan, the customer has to write requirements in their own words and by themselves but
- Customers are not willing to do that; it defeats the whole point of acceptance testing.
- If test cases are written by someone else, the customer does not understand them, so tester has to perform the inspections by themselves only.



finsh





“

**“The job of a tester
is to prove that
every developer
makes a mistake.”**



Summary

Let's See What We Learned.



Testing in SDLC



Functional & Non Functional Testing
validates the software system against the functional requirements



Component Testing
performed on each individual component separately



Acceptance Testing
formal testing based on user requirements and function processing



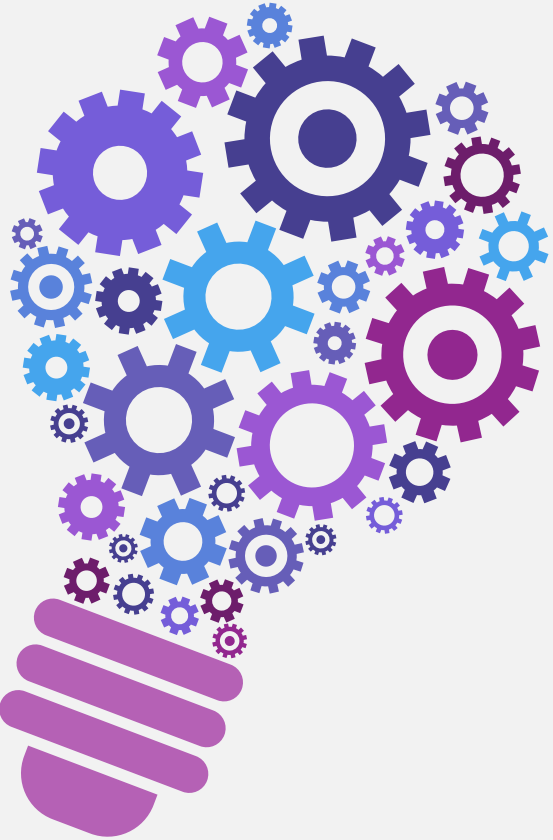
V and W Model
SDLC models where execution of processes happens in a sequential manner



System Testing
Testing the fully integrated applications including external peripherals



Integration Testing
individual units are combined and tested to verify if they are working together



Thanks

Any
questions?

